Bachelorarbeit

Boris Klengel

Neukonzeption des Usability-Labors

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Boris Klengel

Neukonzeption des Usability - Labors

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
Im Studiengang Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. rer.nat. Jörg Raasch
Zweitgutachter : Prof. Dr. Olaf Zukunft

Abgegeben am 20. Juni 2008

Boris Klengel

Thema der Bachelorarbeit

Neukonzeption des Usability - Labors

Stichworte

Gebrauchsfähigkeit, Arbeitsabläufe , Mausmetriken

Kurzzusammenfassung

Das Usability–Labor im Department Informatik der HAW Hamburg wurde im Spätsommer 2007 modernisiert. Aus diesem Grund wurde der gesamte Aufbau des Labors neu entwickelt und die Arbeitsabläufe verändert.
Weiterhin wurde zum einen die im Labor vorhandene Software weiterentwickelt, zum anderen aber auch neue Werkzeuge entwickelt, die die Arbeitsabläufe im Labor verbessern.

Diese Arbeit beschreibt das Ergebnis dieser Neustrukturierung und gibt zudem eine Anleitung für die Nutzer des Labors.

Boris Klengel

Title of the paper

Redesign of the usability laboratory

Keywords

Usability, Workflows, Metrics of mouse movements

Abstract

The usability laboratory at the Department of Computer Science of the Hamburg University of Applied Sciences has been updated in late summer 2007. For this reason the total design of the laboratory was redeveloped and the workflows were changed. Further the software that is available in the laboratory was advanced and new tools were designed which enhance the workflows in the laboratory.

This paper describes the result of the redesign. In addition it issues instructions for the user of the laboratory.

Neukonzeption des Usability – Labors
Bachelorarbeit von Boris Klengel, Juni 2008

Inhaltsverzeichnis

Kapitel 1 – Einleitung und Motivation

Das Usability – Labor im Department Informatik der HAW Hamburg wurde im Spätsommer 2007 grundlegend erneuert. Dabei wurde die alte Hardware durch neue, modernere Technologien ersetzt und ergänzt. Aus diesem Grund musste der gesamte Aufbau des Labors neu entwickelt werden.
Von den Umbaumassnahmen waren auch die einzelnen Workflows betroffen.
Die Arbeitsschritte, die es vor dem Umbau gegeben hat, können nun nicht mehr in der gewohnten Weise durchgeführt werden und müssen den neuen Gegebenheiten angepasst werden. Dies zieht eine Reihe von Veränderungen mit sich nach.
Gleiches gilt für die bereits in früheren Zeiten im Usability – Labor entwickelten Programme, die diese Workflows bewerkstelligen.

Primäres Ziel dieser Arbeit soll es sein, das Ergebnis dieser Neustrukturierung zu dokumentieren und eine Anleitung für die Nutzer des Labors zu geben.

Als weiteres Ziel ist die Weiterentwicklung der im Labor vorhandenen Software sowie die Entwicklung von weiteren Auswertungstools vorgesehen.
Diese Tools sollen die Arbeitsabläufe im Labor verbessern und den Nutzern eine zusätzliche Unterstützung für ihre Arbeit sein. So gab es beispielsweise bisher kaum Tools für die Auswertung und konkrete Weiterverwendung der gesammelten Daten des im Labor verfügbaren Mousetrackers. Der Mousetracker ist ein Programm, das Metriken der Mausbewegungen und Mausaktionen wie X – und Y – Positionen, Anzahl Mausklicks mit den einzelnen Maustasten und die zurückgelegte Strecke mit der Maus (gemessen in Metern) während einer Testsitzung in halbsekündlichen Abständen aufzeichnet und protokolliert. Aus diesen Daten können zwar Diagramme erstellt werden, diese geben jedoch keinen Einblick auf interessantere Details wie Mausverläufe oder Verbleiben der Maus an bestimmten Positionen sowie Dauer dieses „Innehaltens“ wieder. Gerade aus solchen Aspekten könnten Rückschlüsse auf Intentionen und Gedanken der Probanden gezogen werden.
Der im Rahmen dieser Arbeit entwickelte Mousetracer soll da Abhilfe leisten.

Diese Arbeit ist wie folgt gegliedert:
In Kapitel 1 geht es zunächst darum, den allgemeinen Begriff des Usability – Engineerings zu erläutern und abzugrenzen. Der Leser soll verstehen können, worum es in dieser Arbeit geht. Im Anschluss daran folgt eine kurze Beschreibung von Sinn und Zweck des Usability – Labors. Dann wird der hardwaretechnische Aufbau des Labors vor sowie nach dem Umbau erklärt. Für das Verständnis notwendige Darstellungen von Funktionsweisen und Abläufen werden hier ebenfalls gegeben.

Eine vollständige Untersuchung eines Testobjekts im Labor inklusive Betreuung ist in mehreren Workflows gegliedert. Kapitel 2 gibt zunächst eine Auflistung des Ist – Zustandes vor dem Umbau, danach die durch die Umbaumassnahmen veränderten Workflows. Diese Workflows können in naher Zukunft noch vereinfacht werden. Das dritte Unterkapitel skizziert eine Vision dessen, was als bestmöglichster Zustand bzgl. der einzelnen Workflows und des damit verbundenen Arbeitsaufwandes angesehen wird. Damit sich dieser Vision Stück für Stück angenähert werden kann, sind verschiedene Anforderungen zu erfüllen. Diese Anforderungen sind im abschliessenden Abschnitt des 2.Kapitels erläutert und in Prioritäten aufgeteilt.

Zu den Anforderungen müssen geeignete Konzepte formuliert werden, die dann zu gegebenen Zeiten umgesetzt werden können. In Kapitel 3 werden für jede einzelne Anforderung die entsprechenden Konzepte vorgestellt.

Kapitel 4 beschreibt die bisher vorgenommenen Umsetzungen der Konzepte aus Kapitel 3 in konkrete Anwendungen, so wie sie im Rahmen der verfügbaren Zeit möglich waren.

Die realisierten Anwendungen müssen daraufhin einigen Funktionstest unterzogen werden. Kapitel 5 gibt einen Einblick auf die vorgenommenen Tests der in Kapitel 4 beschriebenen Realisierungen. Aufgetretene Probleme werden hierbei festgehalten und mögliche Lösungsvorschläge formuliert. Ein besonderes Kapitel stellt der Abschnitt 5.1 dar, der sich mit der Funktionstauglichkeit des Gesamtsystems beschäftigt.

Aus den gewonnenen Erkenntnissen wird schliesslich versucht, abstrakte Regeln zur Vorgehensweise und Methodik zu finden. Einen Überblick darüber liefert das Kapitel 6.

Kapitel 7 beschäftigt sich schliesslich mit Visionen für Erweiterungen und Verbesserungen des Labors.

Die Arbeit selbst wird mit einer kurzen Zusammenfassung und einem Fazit im Kapitel 8 abgeschlossen.

1.1 Usability-Evaluation

In allen Arbeitsbereichen werden heutzutage Anwendungsprogramme als Werkzeuge benutzt. Jedoch wird häufig beklagt, dass diese Systeme gravierende Mängel in ihrer Benutzbarkeit aufweisen. Viele Nutzer kommen mit den Anwendungssystemen nicht zurecht. Der Erwartungskonformität wird in vielen Fällen nicht entsprochen. Von allgemein bekannten und etablierten Standards wird in zunehmendem Masse abgewichen. Hilfesysteme bringen nur selten Erleichterung, da sie häufig unstrukturiert sind und den Hilfesuchenden meist nicht zu dem Punkt führt, wonach eigentlich gesucht wird. Benötigte Informationen werden daher nur selten auf effiziente Weise geliefert.
Dementsprechend leiden Selbstbeschreibungsfähigkeit und Lernförderlichkeit der Systeme. Zudem wird die Aufgabenangemessenheit insofern oft außer acht gelassen, da der Benutzer seine Aufgaben nur noch auf Umwegen lösen kann.

Hier setzt die Usability - Evaluation an, die im Folgenden beschrieben wird.

1.1.1 Was ist Usability ?

Usability (zusammengesetzt aus dem Englischen, to use = benutzen, gebrauchen, verwenden sowie the ability = die Möglichkeit) meint zunächst einmal grob formuliert die Benutzerfreundlichkeit und die Gebrauchstauglichkeit von Software.
Dies bezeichnet die vom Nutzer erlebte Nutzungsqualität bei der Interaktion mit einem System. Eine besonders einfache, zum Nutzer und seinen Aufgaben passende Bedienung wird dabei als benutzerfreundlich angesehen.

Die Benutzerfreundlichkeit ist eng verbunden mit der Ergonomie. Während man mit Hardware-Ergonomie die Anpassung der Werkzeuge an den Bewegungs- und Wahrnehmungsapparat des Menschen versteht (z. B. Körperkräfte und Bewegungsräume), befasst sich die Software-Ergonomie mit der Anpassung an die kognitiven und physischen Fähigkeiten bzw. Eigenschaften des Menschen, also seine Möglichkeiten zur Verarbeitung von Informationen (z. B. Komplexität) aber auch softwaregesteuerten Merkmalen der Darstellung (z. B. Farben und Schriftgrößen).

Ziel ist dabei die Berücksichtigung des Menschen und seiner Aufgaben und Fähigkeiten sowie die Anpassung des Werkzeuges (sei es Software oder aber jedes andere Werkzeug) daran [Wandmacher 1993].

Wie jedoch kann die Gebrauchstauglichkeit gemessen werden ?
Die Beurteilung der Benutzbarkeit eines konkreten, mit Mängeln behafteten Systems ist eine schwierige Aufgabe. Sie ist nicht durch Betrachtung von Systemeigenschaften erkennbar und messbar, sondern erschließt sich erst in der Benutzung und in den Problemen, die Benutzer mit dem Softwaresystem haben.

In Usability – Tests werden typische Benutzer mit ebenfalls typischen Aufgaben mit dem zu evaluierenden Produkt (ein Artefakt wie eine Software, eine Website, interaktive Geräte und Werkzeuge) konfrontiert und bei der Benutzung systematisch beobachtet. Der Benutzer wird dabei bei seinen Gehversuchen mit dem zu untersuchenden System beobachtet, aus mehreren Perspektiven synchron gefilmt, die Sprache wird aufgenommen und es wird protokolliert, was als Reaktion von Eingaben des Nutzers im Rechner passiert.

Der Usability – Test gilt im Allgemeinen als eine der Referenzmethoden für die Evaluation der Usability. Ihm wird in der Literatur bescheinigt, dass er bereits bei ca. 7 - 8 Nutzern (abhängig von der Komplexität des Testgegenstandes) 80 % der in einem Produkt enthaltenen Probleme identifizieren kann [Nielsen 1994].
Danach fängt der Zyklus an, sich zu wiederholen, und es kommen keine neuen Erkenntnisse mehr hinzu.
Jedoch kann dieses nur als repräsentativ gelten, da Usability – Tests nicht auf Quantität ausgelegt sind. Es kommen Störfaktoren hinzu, so dass gelegentlich Tests auch aufgrund äusserer Einflüsse wenig aussagefähig sind. Es gilt, sich ein Zwischenziel zu merken, man ist möglicherweise demotiviert, hat schlechte Laune, steht unter Stress, ist müde etc. [Graf 2006]

Im Mittelpunkt jeder Evaluation stehen häufig die Kriterien der Gebrauchstauglichkeit nach DIN EN ISO 9241-110 wie Aufgabenangemessenheit, Fehlertoleranz, Selbstbeschreibungsfähigkeit, Erwartungskonformität oder Lernförderlichkeit.
Dadurch wird umrissen, was unter dem Begriff der Software-Ergonomie zu verstehen ist.

„Usability ist der Grad an Qualität, in welchem der Benutzer die Interaktion mit etwas erlebt." [Nielsen 1994]

1.1.2 Das Labor im Department Informatik

Usability – Tests können nicht einfach so durchgeführt werden, dass die Benutzerfreundlichkeit einer Testumgebung dadurch untersucht wird, mehreren Testpersonen Aufgaben bezüglich dieser Umgebung zu geben, die sie dann zu bewältigen haben. Auch wenn schon dort Defizite der Software erkennbar sind, ein blosses Danebensitzen und Zuschauen der Testperson verschafft dem Untersuchenden keinerlei Möglichkeit, solche Tests in geeigneter Form auszuwerten. In Live – Beobachtungen gehen viele Details verloren, da der Mensch nicht fähig ist, sich sämtliche Begebenheiten innerhalb kürzester Zeit merken zu können. Protokollhafte Aufzeichnungen scheinen auf dem ersten Blick hilfreich, doch dies ist irreführend. Denn während der Testleiter schreibt, passieren Dinge, die er dann nicht mitbekommt oder die ihn ablenken und eigentlich beabsichtigte Notizen vergessen lassen. Auch eine Wiederholung des Geschehens ist unmöglich. Man kann nicht einfach hingehen und dem Probanden sagen, die eben vorgenommenen Aktivitäten in genau derselben Weise noch einmal durchzuführen. Die Gefühle des Probanden und Beobachtungen während der ursprünglich und erstmaligen Auftreten von Problemen oder Situationen lassen sich nicht rekapitulieren und sind somit verloren.

Eine Aufzeichnung des Geschehens mit modernster Technik vermeidet diese Problematik. Durch Aufnahme von Bild und Ton können Videos des Tests erstellt werden, die sich später immer und immer wieder abspielen lassen. Dadurch erhält der Testleiter eine Möglichkeit, den Test besser auswertbar zu machen, da das Video den Test genauso wiederholen lässt, wie er im Original statt gefunden hat. Weiterer Vorteil ist, dass dem Testleiter oft erst später bei iterativer Ansicht der Aufzeichnungen immer wieder neue Details zugänglich werden, die ihm während des eigentlichen Tests gar nicht aufgefallen sind.

Um eine solche Auswertungsweise zu ermöglichen, benötigt man eine spezielle Laborumgebung. Genau diesem Zweck soll das Usability – Labor der HAW Hamburg dienen. Erstmalig eingerichtet im Jahr 2003 hat es sich im Laufe der Zeit weiterentwickelt und hat sich inzwischen zu einem festen Bestandteil der Informatikausbildung etabliert. Es wird vor allem in Lehrveranstaltungen wie Projektstudien zum Thema Usability und dem Fach Software – Engineering eingesetzt.
Aber auch Untersuchungen von Diplom –, Bachelor – oder Masterarbeiten sowie externen Projekten von Firmen werden hier in zunehmenden Masse durchgeführt.

Hauptzielgruppe sind daher in erster Linie die Studenten des Departments Informatik, die dadurch eine erhebliche Kompetenz und Sensibilität in ergonomischer Richtung erhalten sollen. Idealziel ist, dass jeder Studierende im Laufe seines Studiums mit seinen entworfenen Systemen mehrmals im Usability-Labor erscheinen soll, wo er seine Arbeiten überprüfen kann.

Das Labor selbst ist in zwei Bereiche aufgeteilt. Im Testraum findet die eigentliche Untersuchung des zu überprüfenden Objekts statt. Dies können beispielsweise kleinere Anwendungen, grosse Systeme oder auch Webseiten sein. Sofern notwendig, werden dabei die Anwendungen auf dem verfügbaren Testrechner vor den eigentlichen Testdurchläufen installiert.
Im Testraum arbeitet der Benutzer am Objekt, indem er von dem Untersuchenden (oftmals ist dies gleichzeitig der Entwickler der Anwendung) gestellte Aufgaben zu bewältigen hat. Dabei wird er aus verschiedenen Kameraperspektiven gefilmt.
Seine sprachlichen Emotionen werden tontechnisch aufgezeichnet.
Hierbei spielt das „Laut – Denken – Protokoll“ eine tragende Rolle. Der Benutzer soll seine Gefühle sowie positive und negative Kritik verbal zum Ausdruck bringen, und zwar während des Geschehens, nicht erst im Nachhinein.
Über Mikrofon und Lautsprecher steht der Proband die ganze Zeit über mit dem Testleiter in Verbindung, so dass dieser zu jeder Zeit – ganz besonders in kritischen Situationen – von aussen eingreifen und der Testperson die nötige Hilfestellung geben kann. Ein Eindruck vom Testraum kann aus der nachstehenden Abbildung 1 – 1 – 1) gewonnen werden.



Abb. 1-1-1: Blick in den Testraum

Im Regieraum hält sich der Testleiter auf und beobachtet die Ereignisse über die Kameramonitore. Dazu werden Notizen gemacht, die er dann später für seine Auswertung verwertet. Mit einigen eigens entwickelten Anwendungen, die dem Testleiter bei den Testdurchgängen zur Verfügung stehen, kann der Testleiter weitere, wertvolle Informationen festhalten, die für die Auswertung möglicherweise von Interesse und Bedeutung sind und diese erleichtern können.
Alle Informationsströme laufen hier zusammen und werden später zu einem multimedialen Protokolldokument zusammengeschnitten. Abbildung 1 – 1 – 2 zeigt einen Blick in den Regieraum.



Abb. 1-1-2: Blick in den Regieraum

1.2 Motivation

Bisherige Untersuchungen im Usability-Labor zeichneten sich dadurch aus, dass besonders die Nachbereitungen und Auswertungen von Tests aufwendig und zeitintensiv waren, und es nur wenige Möglichkeiten zur Automatisierung von Arbeitsabläufen gab. Dies soll nun geändert werden. Dazu wird das vorhandene System im Labor durch ein neueres und besseres ersetzt, was grössere Änderungen in der Struktur und im Aufbau des Labors nach sich zieht.

Dieses Kapitel beschreibt zunächst einmal den prinzipiellen Aufbau des Labors vor dem grossen Umbau. Die Neukonzeption sowie die neu vorhandenen Technologien und die damit verbundenen Absichten werden darauf im Anschluss vorgestellt.

1.2.1 Das Labor vor dem Umbau

Zuallererst ist zu sagen, dass das Usability – Labor über ein eigenes Netzwerk verfügt, das nicht an das Netz der HAW – Hamburg angeschlossen ist. Entsprechende Leitungen und Buchsen sind zwar gelegt und verfügbar, werden aber bislang nicht genutzt.

Wie in der Abbildung 1 – 2 – 1 zu sehen ist, verfügte der Regieraum über zahlreiche Komponenten und Geräte, die für den Empfang und die Verarbeitung aller eingehenden Video – und Audiodaten zuständig waren (vgl. auch [Schwarze 2003]).

Die erste Einheit zum Empfang der Videosignale bildeten die damaligen vier Kameramonitore, welche das Livebild aus dem Testraum anzeigen. Dies ist in etwa vergleichbar mit einer Videoüberwachung wie zum Beispiel in Kaufhäusern oder auf Tankstellen.
Von dort aus wurden diese Streams weitergeleitet an ein System, das aus sechs einzelnen Festplattenrecordern bestand. Für jede der Aufnahmequellen (Kameras 1 bis 4, VGA – Bild des Testrechners sowie Aufzeichnung der Metriken) stand eine eigene Festplatte zur Verfügung, auf die über das Netzwerk zugegriffen werden konnte. Alle Festplatten waren im Netzwerk unter einem eigenen Laufwerksbuchstaben geführt, den Bezeichnungen von N:\ startend und dem Alphabet absteigend bis zum Buchstaben S:\ fortzählend.
Im Gegensatz zum heutigen System hatte dies den gravierenden Nachteil, dass eine Festplatte nicht weiterbenutzt werden konnte und ausgetauscht werden musste, wenn Probleme und Ausfälle auftraten.
Die Kameraaufzeichnungen wurden damals noch direkt im AVI – Format auf den Festplattenrecordern gespeichert.
Das VGA – Bild wurde vom Testrechner aus gesplittet (aufgeteilt) und einerseits auf dem zugehörigen Monitor des Testrechners in herkömmlicher Weise ausgegeben, andererseits zu einem zweiten, im Regieraum befindlichen Monitor weitergeleitet.
Die für die Aufzeichnung benötigten Streams wurden dann direkt von dem VGA – Monitor im Regieraum abgegriffen und auf die dafür vorgesehene Festplatte geschrieben.

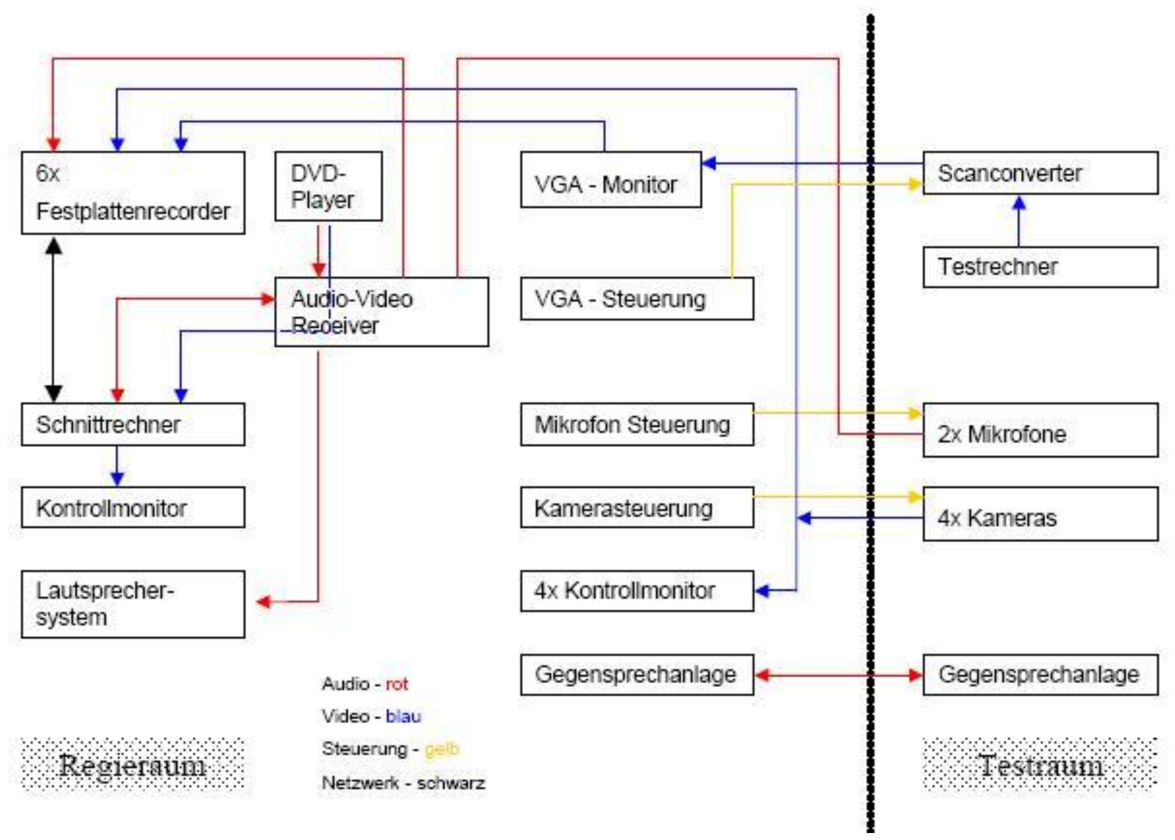


Abb. 1-2-1: Komponenten und Verbindungen im Usability – Labor vor Umbau

Zu erwähnen sei, dass der VGA – Monitor auf der Regieseite aus zwei Bildschirmen bestand, wie dies auch heute noch der Fall ist. Der Grund hierfür ist, dass der Proband im Testraum nichts vom Mousecapturer mitbekommen soll. Dementsprechend wurde der Bildschirm des Testrechners auf einen erweiterten Windows – Desktop eingestellt. So können bestimmte Anwendungen und Oberflächen aus dem Bildschirmbereich gezogen werden, so dass der Proband diese nicht sehen kann, jedoch der Testleiter über den zweiten angeschlossenen Bildschirm.

Der im Testraum hörbare Ton wurde über 2 aus der Decke ragenden Mikrofone eingefangen, über einen Audio – Video – Empfänger verstärkt mit Hilfe der Lautsprecher direkt im Regieraum wiedergegeben und zudem zusammen mit dem Stream von Kamera 1 auf dem entsprechenden Festplattenrecorder gespeichert.
Die Mikrofone selbst sind nach wie vor genauso wie die einzelnen Kameras über Fernbedienungen vom Regieraum aus steuerbar und einstellbar.

Ein für die Aufbereitung bereitgestellter sogenannter Schnittrechner ist über LAN mit den Festplattenrecordern verbunden und gewährt somit direkten Zugriff auf die gespeicherten Aufnahmen. Die einzelnen Video – und Audiodatenströme werden hier mit dem XviD – Codec komprimiert und zu einem Videoclip zusammengefügt [XviD 2006].

Für den synchronen Start mehrerer Prozesse auf den verschiedenen Rechnern sorgte ein eigens programmierter Zeitserver, der beim Hochfahren des Schnittrechners automatisch gestartet wurde.

Der Testraum ist zunächst einmal mit einem gewöhnlichen Arbeitsplatz ausgestattet, bestehend aus einem handelsüblichen PC. Hierauf laufen die zu testenden Anwendungen ab. Wie angedeutet beinhaltet jedoch der Bildschirm des Arbeitsplatzes nur die eine Hälfte des gesplitteten Gesamtbildes. Der PC ist demnach so konfiguriert, dass zwei Monitore an dem Gerät angeschlossen sind und sich die Anzeige teilen, aber nur ein Monitor tatsächlich an dem Arbeitsplatz steht (der zweite Bildschirm ist der Metriken – Bildschirm im Regieraum).

Weiter ist in diesem Raum an allen vier Ecken jeweils eine Kamera angebracht, so wie sie auch in der Überwachungstechnik verwendet werden. Diese Kameras zeichnen das Geschehen am Arbeitsplatz analog auf und senden diese Bilder weiter an die Monitore im Regieraum. Die Verbindung besteht dabei aus S – Video–Kabeln.

S – Video (auch bekannt als Separate Video, Y/C) bezeichnet das getrennte Übertragen von Helligkeit – und Farbinformationen, mit entsprechend ausgeführten Kabel- und Steckverbindungen. Es ermöglicht ein qualitativ hochwertiges Signal, erreicht jedoch nicht die Qualität von RGB – Formen.
Für das in Deutschland gebräuchliche PAL – Format wird das Farbsignal bei 4,43361875 MHz übertragen. Die Bandbreite der Farbartmodulation beträgt nominal etwa ± 1,3 MHz.

S – Video ist kein Videoformat. Es handelt sich hierbei eher um einen Übertragungsvorgang von analogen Daten auf Basis der allgemeinen Fernsehnorm. Je nach Übertragungsnorm können Helligkeit – und Farbsignal andere Frequenzen belegen. Sie werden jeweils über zwei separate Leitungspaare übertragen.

Test – und Regieraum waren über eine beidseitige Gegensprechanlage verbunden, so dass Testleiter und Proband auch darüber miteinander kommunizieren konnten.

Weiterhin befinden sich im Testraum ein Windows Mobile PDA – Gerät
sowie ein Netzwerkrouter, über den die Kommunikation zwischen Testrechner und Schnittrechner ablief, und an dem ein weiterer Rechner (der für die Verbindung zum PDA über WLAN zuständig ist) sowie ein Notebookanschluss angekoppelt sind. Somit lassen sich auch Notebooks unkompliziert als Testrechner verwenden.

Die nachstehende Abbildung 1 – 2 – 2 zeigt einen groben Aufbau des laborinternen Netzwerkes. Dabei steht der mit der IP – Adresse 10.0.2.2 angegebene Rechner für den Schnittrechner. Der Testrechner ist jener mit der IP – Adresse 10.0.2.3. [Yaylacioglu 2007]

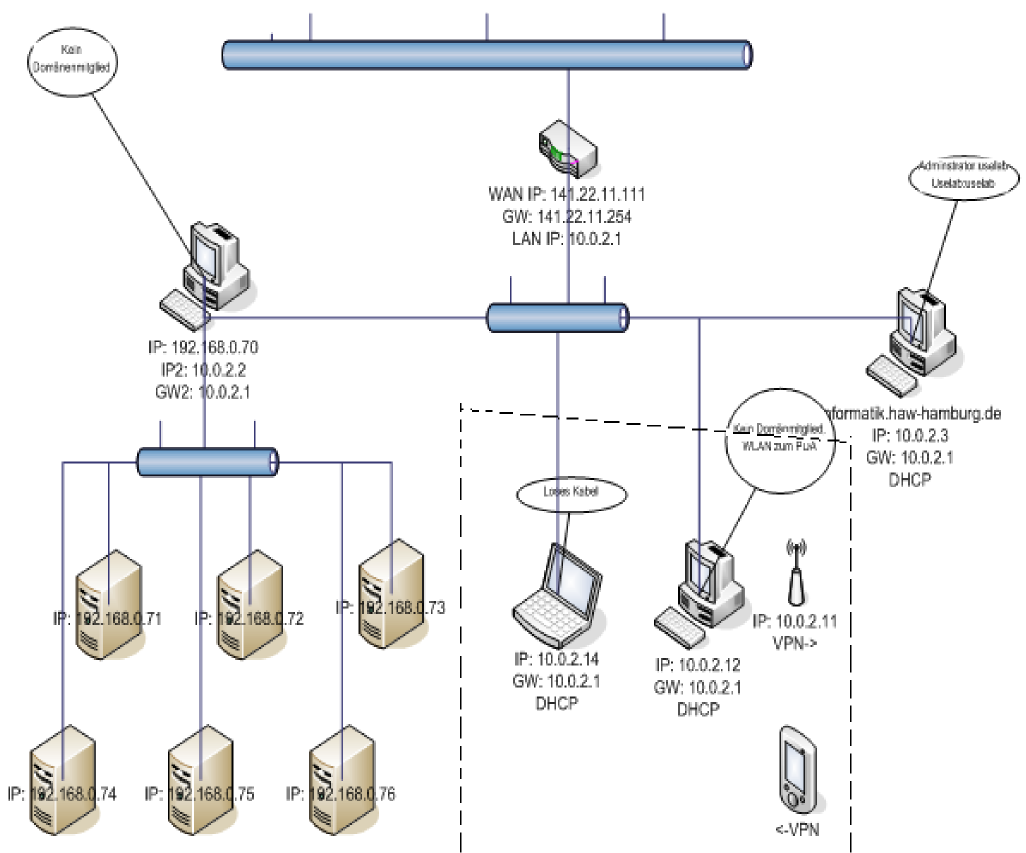


Abb. 1-2-2: Netzwerkaufbau Usability – Labor bis September 2007

Die Rechner sowie das PDA – Gerät (in der Abbildung innerhalb der gestrichelten Linie) wurden nur für andere Bachelorarbeiten eingerichtet und gehörten somit nicht zur direkten Ausstattung des Labors.

1.2.2 Die neue Ausstattung und damit verbundene Ziele

Im September 2007 wurde das Labor in den Räumen der HAW Hamburg grundlegend erneuert. Die Ausstattung wurde komplett modernisiert, woraus sich einige Änderungen bezüglich des Netzwerkaufbaus, aber auch der internen Abläufe („Workflows“) ergeben.

Die vier bereits vorhandenen Kameras im Testraum wurden beibehalten, deren Arbeitsweise hat sich nicht verändert. Ergänzt wurden sie durch zwei weitere Kameras, die sich jedoch nicht gravierend von den alten unterscheiden. Auch sie sind so eingestellt, dass sie ein S - Video - Signal senden (obwohl auch andere Aufnahmearten möglich sind), damit sie bestmöglich zu den alten Kameras passen und die aufgenommenen Bilder sich nicht voneinander unterscheiden. Wie bei den anderen Kameras werden die Bilder analog zu zwei weiteren Monitoren im Regieraum übertragen.

Diese Kameras sind so angebracht, dass sie den Probanden von vorn aufnehmen. Dadurch können weitere Rückschlüsse aus den Empfindungen der Testperson gezogen werden, bislang waren Gesichtsausdrücke nur schwer erkennbar. Nachteil ist, dass die Kameras direkt im Blickfeld des Probanden angebracht sind, was ihn möglicherweise verstören oder in Aufregung versetzen könnte, da Menschen im allgemeinen nicht gern beobachtet werden. Es hat sich jedoch gezeigt, dass auch diese Kameras recht schnell „vergessen" werden.

Der Testrechner wurde durch einen neueren, leistungsstärkeren ersetzt.
Auf ihm wurden zwei Benutzerkonten angelegt. Dies ist zum einen ein passwortgeschütztes Konto für den Administrator bzw. Operator, der einen vollen Zugriff auf alle Einstellungen und Programme benötigt. Notwendige Installationen finden unter diesem Konto statt, der Administrator muss also während einer Installation einer Testumgebung immer anwesend sein.
Das zweite Konto „como" - nicht passwortgeschützt - ist als „Gastkonto" gedacht. Hier können alle Programme ausgeführt, jedoch nicht verändert werden. Sofern nicht anders abgesprochen, laufen die meisten Tests unter diesem Konto ab, es sei denn, die Testperson benötigt Administratorrechte.
Die Software auf dem Testrechner ist dieselbe wie auf dem älteren Rechner, in der Hinsicht ist also alles beim Alten geblieben.

Ergänzt wurde dieser um einen zweiten Rechner, der zudem an das interne HAW - Netz angeschlossen ist. Zugänglich ist er mit jedem gewöhnlichen Studentenkonto (Benutzername und Passwort, wie in den anderen Laboren).
Somit haben insbesondere für Lehrveranstaltungen wie Software - Engineering Studenten die Möglichkeit, ihre Software direkt aus ihrem
Public - Verzeichnis zu testen, ohne dass diese auf dem Testrechner installiert werden muss. Aus Sicherheitsgründen dürfen auf diesem Rechner jedoch keine Installationen getätigt werden, um keinen Schaden im HAW - Netz anrichten zu können. Daraus ergibt sich der Nachteil, dass zusätzliche, informationsliefernde Programme wie Mousecapturer hier nicht ausgeführt werden können. Insofern ist dieser zweite Testrechner engen Restriktionen in der Nutzung unterworfen.

Neu hinzugekommen ist ein Gerät, das sich „Eyetracker" nennt. Mit diesem Gerät ist es möglich, den Blickverlauf der Testperson aufzuzeichnen und auszuwerten. Man kann u.a. nachweisen, wo die Testperson zu jedem Zeitpunkt während des Tests hingeschaut hat und dementsprechend die Bereiche lokalisieren, in denen die persönliche Aufmerksamkeit des Benutzers lag. Dies kann beispielsweise für Untersuchungen von Webseiten von Interesse sein, wenn es darum geht, herauszufinden, inwieweit Werbebanner den einzelnen Personen auffallen.
Der Eyetracker befindet sich derzeit noch selbst in der Untersuchungsphase und ist Gegenstand einer anderen Diplomarbeit von Hr. Stefan Richter.
Nähere Einzelheiten sind dort zu entnehmen.
Dieses Gerät ist als eigenständige Einheit an das laborinterne Netzwerk
angeschlossen, so dass die zugehörige Software vom Schnittrechner aus ferngesteuert gestartet werden kann.

Grössere Veränderungen hat es im Regieraum gegeben.
Zunächst sind hier zwei weitere Monitore hinzugekommen, die dem Testleiter die Aufzeichnungen der beiden neuen Kameras wiedergegeben. Insgesamt sind also jetzt sechs Kameramonitore vorhanden, aufgestellt in zwei übereinander liegenden Zeilen zu je 3 Monitoren (siehe Abbildung 1 – 2 – 3).
Nach wie vor sind die einzelnen Kameras über die Kamerasteuerung einstellbar. Für jede Kamera sind dort zwei feste Positionsangaben bereist eingespeichert. Zum einen ist dies die Ausrichtung während des Testverlaufs (Position 1), zum anderen die während des Nachgesprächs (Position 2).
Position 2 bedeutet, dass die Kamera zum Nachgespräch weiter aufgezogen wird (Aufhebung des Heranzoomens), so dass ein grösseres Blickfeld sichtbar wird.



Abb. 1-2-3: Arbeitsplatz des Testleiters mit Kameramonitoren und Mikrofon

Die Ausstattung für Tonaufnahmen, Tonwiedergabe (Lautsprecher, Verstärker etc.) und VGA - Anzeige ist so geblieben wie vor dem Umbau und muss daher nicht weiter erläutert werden. Lediglich die Gegensprechanlage wurde modifiziert. Sie funktioniert jetzt nur noch in der Richtung vom Regieraum zum Testraum, da eine Kontaktaufnahme des Probanden zum Testleiter durch die Mikrofone im Testraum bereits gegeben ist und eine zusätzliche, rückläufige Gegensprechanlage überflüssig macht.

Die Abbildung 1 – 2 – 4 zeigt den aktuellen Stand des Netzwerks im Usability – Labor.

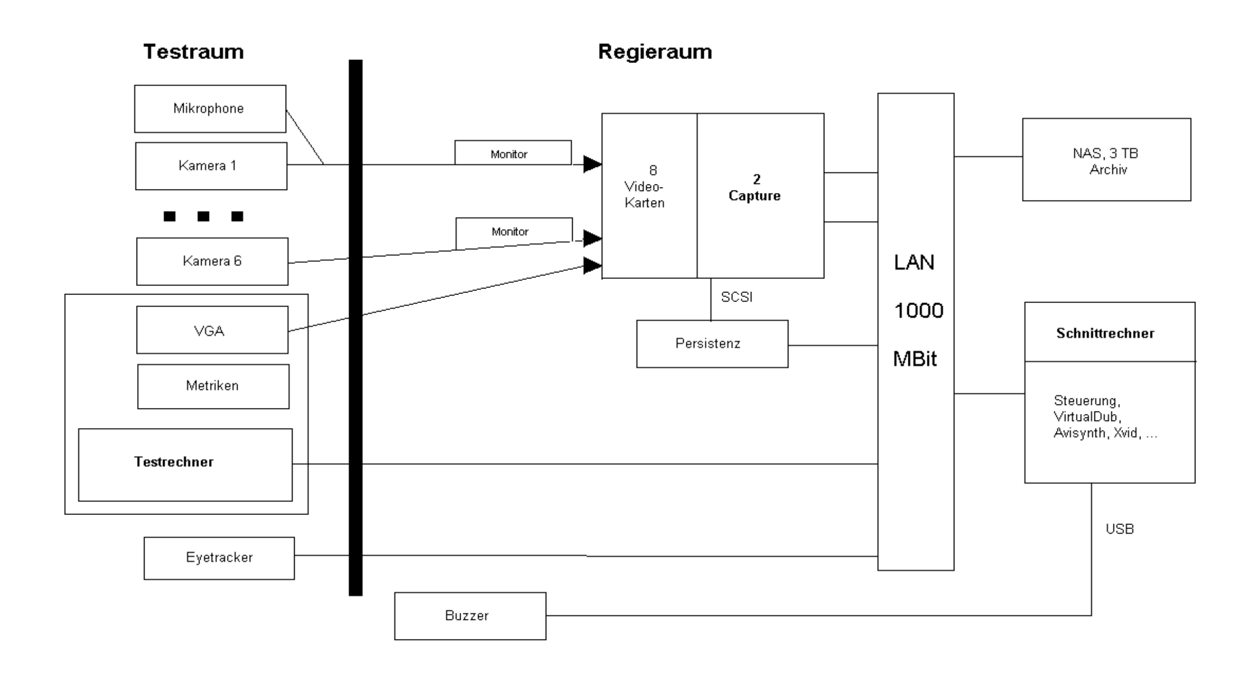


Abb. 1-2-4: Komponenten und Verbindungen im Usability - Labor (aktuell)

Die Kamerabilder sowie das VGA - Bild werden von hier aus weitergeleitet zu zwei Videograbbern (im Folgenden auch als Capturerechner bezeichnet), die für die Speicherung und Umwandlung der Live - Bilder in Videoclips zuständig sind. Die Filme werden jetzt mit einer Framerate von 25 Bildern pro Sekunde aufgenommen und als MPEG4 - Stream weggeschrieben. Dies ist im Vergleich des im alten System benutzten Verfahrens (AVI) ein höheres Mass an Datenkompression. Die einzelnen Kameraaufzeichnungen verbrauchen nicht mehr so viel Speicher wie in früheren Zeiten.
Zu diesem Zweck ist jeder der Videograbber mit je vier Videokarten ausgestattet, jeweils eine Karte für eine Kamera. Die Aufteilung sieht derzeit wie folgt aus:

Capturerechner 1:
Eingang 1 → Kamera 1, zusätzlich wird hier das Tonsignal empfangen
Eingang 2 → Kamera 2
Eingang 3 → Kamera 3
Eingang 4 → Kamera 4

Capturerechner 2:
Eingang 5 → nicht belegt
Eingang 6 → Kamera 6
Eingang 7 → VGA - Bild
Eingang 8 → Kamera 5

Alle Daten werden von beiden Grabbern auf eine externe Festplatte geschrieben, die in mehreren einzelnen Partitionen aufgeteilt ist. Betrieben wird diese Festplatte im RAID 5 - Modus, was bedeutet, dass eine Partition ausfallen darf, ohne den Betrieb einzuschränken.

Die Filmaufbereitung passiert wie gewohnt auf einem Schnittrechner, der - ähnlich wie der Testrechner - durch einen neueren, leistungsstärkeren ersetzt wurde. Damit wird das Ziel verfolgt, durch einen grösseren Speicher mehr Filme unterbringen und durch eine verbesserte Performanz Filme schneller zusammenstellen zu können als bisher.

Schnittrechner und Capturerechner sind durch einen Switch verknüpft, so dass alle drei Rechner über eine einzige Tastatur, eine Maus und einen zweigeteilten Bildschirm bedienbar sind. Mit der Tastatur kann zwischen den einzelnen Rechnern hin - und her geschaltet werden.
Für den Zugang zu diesen Rechnern wird ein Passwort benötigt. Momentan sind auf allen drei Rechnern dasselbe Benutzerkonto mit Benutzername und Passwort eingerichtet. Diese Daten sind nur den Verantwortlichen bekannt.

Zusätzlich wurde ein NAS – Speicher (im folgenden auch als Linkstation bezeichnet), ein so genannter Buffalo, beschafft. Dieser externer Speicher ist für Datenhaltung zuständig und wird nur zu Archivierungszwecken und Backup - Erstellung eingesetzt. Backups sind von unschätzbarem Wert, da im Falle eines gravierenden Systemausfalls zerstörte Daten auf einfache Weise wiederhergestellt werden können.
Die Linkstation hat eine Speicherkapazität von 3 Terabyte. Dadurch wird das Problem der Speicherplatzbedarfs umgangen, denn es dürfte Jahre dauern, bis diese Kapazität erschöpft ist.

Zusammengehalten werden die Einzelteile im Regieraum (Capturerechner, Schnittrechner und Linkstation) durch ein Netzwerk - Switch, an dem auch der Probandenrechner über den Router (im Testraum) angeschlossen ist.
Somit entsteht ein neu aufgezogenes Netzwerk, das folgende Bestandteile enthält:

- Netzwerk - Router: IP 10.0.2.1
- Testrechner: IP 10.0.2.3
- Eyetracker: IP 10.0.2.26
- Gastanschluss für Notebooks: IP 10.0.2.14
- Capturerechner 1: IP 10.0.2.19
- Capturerechner 2: IP 10.0.2.21
- Schnittrechner: IP 10.0.2.22



Abb. 1-2-5: Rechnerturm im Regieraum

Abbildung 1 – 2 – 5 zeigt den im Regieraum stehenden Rechnerturm, in dem sich (in der Abbildung von oben nach unten) die Linkstation, die beiden Capturerechner, der Schnittrechner sowie die externe Festplatte und der Netzwerk-Switch befinden.

Eine neuere Errungenschaft ist auch das Buzzertool [Zlotopolski 2006], eine Metallbox, an der kleinere Knöpfe sowie zwei grössere, aus Quizshows bekannte Buzzer befestigt sind. Es erlaubt dem Testleiter, zusätzliche Informationen im Verlauf eines Tests zu sammeln (Abbildung 1 – 2 – 6).
Diese Box ist über einfache Chinch-Kabel an den Schnittrechner angeschlossen und nimmt Tonaufnahmen des Testleiters sowie Kopien des VGA - Bildschirms in Form von Screenshots entgegen. Nähere Einzelheiten hierzu sind in den folgenden Kapiteln dokumentiert.



Abb. 1-2-6: Buzzertool mit VGA – und Metrikenmonitor

Kapitel 2 – Vision

Die Durchführung eines kompletten Usability-Tests ist mit einer Reihe von Workflows verbunden. Im Laufe der Zeit haben sich diese Workflows verändert. Dieses Kapitel beschreibt einerseits den Ist-Zustand vor dem Umbau, andererseits den Ist-Zustand mitsamt den Anpassungen nach dem Umbau. Der Soll-Zustand, der in der Zukunft erreicht werden soll, wird in Kapitel 2.4 erläutert. Zur Erlangung des Soll-Zustandes sind noch viele Anpassungen vorzunehmen und weitere Tools zu entwickeln. Einige dieser Ergänzungsmöglichkeiten werden in Kapitel 2.5 im Rahmen einer Anforderungsplanung vorgestellt, welche ein erster Schritt in Richtung der in Kapitel 2.4 beschriebenen Visionen darstellen soll.

2.1 Methodik der Durchführung eines Usability - Tests

Im Folgenden wird kurz beschrieben, wie ein Usability – Test durchgeführt wird.
Die darin enthaltenden Arbeitsschritte des Testleiters haben sich auch nach der Neukonzeption nicht verändert.

In einem ersten Schritt muss sich der Testleiter mit dem zu Testobjekt auseinandersetzen und gezielt nach Schwachstellen suchen. Hierzu werden Hypothesen entwickelt, an welchen Stellen Nutzer Probleme mit der zu untersuchenden Software haben könnten. Um die Vermutungen nachweisen zu können, werden daraus Aufgabenstellungen abgeleitet, die Testpersonen während der Durchführung eines Tests bearbeiten sollen.
Sind die Aufgaben formuliert, beginnt der eigentliche Test. Es muss nun eine geeignete Testperson rekrutiert, in das Labor gebracht und eingewiesen werden. Um der Testperson eventuelle Angst, Aufregung und Unwohlsein aufgrund der klinischen Atmosphäre und der Beobachtung durch die Kameras zu nehmen, wird sie zu Beginn des Tests in ein lockeres Gespräch verwickelt. In diesem Gespräch werden der Testperson einzelne Details des Labors sowie der grundsätzliche Testablauf erklärt.

Dann wird der Testperson die eigentliche Aufgabenstellung näher gebracht und der Aufgabenzettel ausgehändigt. Der Testleiter hat auf Fragen des Probanden einzugehen, was besonders wichtig ist, da Verständnisfragen nicht erst während des Tests aufkommen sollen. Üblicherweise dauert ein einzelner Test im günstigsten Fall etwa 15 bis 20 Minuten. Unklarheiten während der Aufgabenbearbeitung würden den Test nur unnötig in die Länge ziehen und möglicherweise die spätere Auswertung erschweren oder den Testleiter falsche Schlüsse ziehen lassen.
Ausserdem soll jede einzelne Testperson gleich behandelt werden und die Ausgangsposition in jedem Fall identisch sein, da sonst das Testergebnis verfälscht werden könnte.

Im Anschluss an dieses Vorgespräch erfolgt die Bearbeitung der Testaufgaben durch den Probanden.

Nach Erledigung der Aufgaben, spätestens aber nach Ablauf der zur Verfügung gestellten Zeit wird der eigentliche Test durch den Testleiter beendet. Der nächste Schritt besteht darin, mit der Testperson ein Nachgespräch durchzuführen.
In diesem Nachgespräch soll der Proband im Rahmen eines Interviews dem Testleiter seine Eindrücke von dem Testobjekt verdeutlichen und sowohl positive als auch negative Kritik üben. Das Nachgespräch ist eine wichtige Basis für die spätere Auswertung und orientiert sich meist an einen Fragebogen, der von dem Testleiter vor dem Testszenario erstellt wird.
Nach dem Interview wird die Aufzeichnung beendet und die Testperson verabschiedet.

2.2 Workflows vor dem Umbau

Der Testleiter hat während eines gesamten Testszenarios eine Vielzahl an Aufgaben zu erledigen. Die Durchführung eines Tests inklusive der Betreuung der Testperson, wie im vorangegangenen Kapitel beschrieben wurde, ist nur ein Teil der Aufgaben, die der Testleiter im Rahmen eines Tests zu übernehmen hatte.
Dieses Kapitel beschreibt kurz die weiteren Arbeitsabläufe des Testleiters, wie sie vor der Neukonzeption des Labors vorlagen.

Im ersten Schritt musste der Testleiter zunächst einmal das gesamte System einschalten. Hierzu gehörten u.a.

- Hochfahren des Test – sowie des Schnittrechners
- Einschalten der damaligen Festplattenrecorder
- Einschalten aller sonstigen kleineren Geräte wie Mikrosteuerung oder VGA-Splitter
- Ausrichten der vier Kameras
- Installation der zu testenden Software sowie Initialisierung der Testumgebung
- Testen der Gegensprechanlage (Soundcheck) und Grundeinstellung der Lautstärke
- Überprüfung der Speicherkapazität der Festplattenrecorder, da diese damals sehr begrenzt war
- Programm zur Aufnahme der Videodaten starten

Wenn das Testsystem erfolgreich hochgefahren werden konnte und sichergestellt war, dass die benötigte Hardware und Software ordnungsgemäss lief, wurde der eigentliche Usability – Test mit dem Probanden vorgenommen (vgl. Kapitel 2.1).

War das Vorgespräch beendet, wurde der Aufnahmeprozess durch den Testleiter gestartet. Für die Aufzeichnung von Mausdaten wie z.B. Anzahl der Klicks mit der linken Maustaste musste auf dem Testrechner der sogenannte Mousecapturer gestartet und aus dem Bildschirmbereich gezogen werden. Die Clientseite des Mousecapturers war auf dem Schnittrechner zu starten. Die Aufzeichnung der Kameras und des VGA-Bildschirms wurde anschliessend über das auf dem Schnittrechner verfügbare Programm „Keypad“ begonnen und bei Testende wieder gestoppt.

Nach der Testdurchführung musste der Testleiter aus den einzelnen Videoaufnahmen einen einzelnen Videoclip generieren. Hierzu war das Programm „Avsgenerator“ zu starten, das für den Film ein AviSynth-Skript erstellt, was dann als Eingabe für das Videoschnittprogramm VirtualDub diente. Leider konnte zu dem Zeitpunkt jedoch nur ein einzelnes AviSynth-Skript auf einmal erstellt werden, so dass dieser Vorgang für jeden Test zu wiederholen war.
Im nächsten Schritt wurde der Film wie angesprochen über VirtualDub zusammengeschnitten. Im Vergleich zu heute nahm die Filmerstellung viel Zeit in Anspruch. Das Rendern eines einzelnen Films dauerte etwa viermal so lang wie die Kameraaufzeichnung selbst. Bei einem Test von 15 Minuten Länge entspricht dies einer Dauer von einer Stunde, bis der Endfilm vorliegt.

Zum Abschluss des Zyklus wurde der fertige Film manuell gesichert. Zu diesem Zweck gab es auf den Schnittrechner eine eigene Partition, auf die der Film archiviert wurde. War dies erfolgreich erledigt, wurde das komplette System in umgekehrter Reihenfolge heruntergefahren.

Die Auswertungen der Tests wurden meist nicht im Labor vorgenommen. Es gab zu dem Zeitpunkt mit Ausnahme des Tools von Daniel Wolf auch keine geeigneten Auswertungstools, die den Testleiter bei diesem Workflow unterstützen konnten [Wolf 2006].

Diese Workflows wurden während der Neukonzeption teilweise gravierend geändert. Das nachfolgende Kapitel beschreibt daher die aktuellen Arbeitsschritte, die im Labor durchzuführen sind.

2.3 Workflows nach dem Umbau

Zunächst einmal ist festzuhalten, dass jeder Testdurchlauf zukünftig von zwei Personen mit verschiedenen Rollen betreut wird. Zum einen ist da der eigentliche Testleiter zu nennen. Der Testleiter ist weiterhin für die Durchführung der einzelnen Usability-Tests verantwortlich, hat jedoch nun nichts mehr mit der Bedienung des Systems und der verwendeten Programme zu tun. Der Testleiter soll sich voll und ganz auf den Test selbst konzentrieren und nicht mehr wie vorher gleichzeitig mit der Filmaufnahme und der Nachbereitung beschäftigt sein.

Alle vorhandenen Rechner sind passwortgeschützt. Unbefugten soll kein Zugang zu dem System gewährt werden. Die Passwörter sind nur dem neu hinzugekommenen Operator, den betreuenden Professoren oder den Labormitarbeitern bekannt, jedoch nicht laborfremden Personen. Hierzu gehört auch der Testleiter.

Für den Testleiter haben sich die Workflows nicht geändert. Dies betrifft die Begrüssung und Einweisung des Probanden in das Testobjekt, die Beobachtung und Protokollierung der Vorgänge während des Tests sowie Durchführung des Nachgesprächs (vgl. hierzu das vorangegangene Kapitel). Jedoch sollen ihm nun weitere Tools zur Verfügung stehen, die er während der Tests verwenden kann, und die ihm bei der späteren Auswertung behilflich sein sollen.

Die technische Betreuung eines Testszenarios wird durch einen Operator vorgenommen. Ihm fallen Aufgaben wie Hochfahren und Sicherstellen der Funktionsfähigkeit des Systems, Video – und Tonaufnahme sowie Fertigstellen des Videoclips (welches später dem Testleiter ausgehändigt wird) zu.
Aus der Testphase des neuen Gesamtsystems (vgl. Kapitel 5.1) haben sich einige Veränderungen der Workflows für den Operator ergeben. Desweiteren ist der eine oder andere neue Arbeitsschritt hinzugekommen.

1. Installation des Testobjekts, falls erforderlich

   Falls es notwendig ist, die zu testende Anwendung zu installieren (dies ist dann der Fall, wenn der Testleiter in den vollständigen Genuss der Auswertungsmöglichkeiten kommen will, oder es sich bei dem Testsystem um einen Auftrag von einer externen Firma handelt), so hat der Operator vor Start einer Testreihe die Installation durchzuführen oder zumindest zu betreuen, da nur er die nötigen Kenntnisse und Zugangsdaten zu dem System im Labor besitzt.
   In diesen Fällen muss der Testrechner, der keinen Anschluss an das HAW-Netz besitzt, verwendet werden, da auf dem anderen Rechner aus Sicherheitsgründen keine Installationen vorgenommen werden dürfen (dies ist auch der Grund, weshalb auf diesem Rechner auch Auswertungstools wie der Mousecapturer nicht verfügbar sind).

   Es hat sich bewährt, die Installation bereits mindestens einen Tag vor der geplanten Untersuchung vorzunehmen, um Luft für die Behebung eventuell auftretender technischer Schwierigkeiten zu lassen.

2. Starten des Gesamtsystems

   Vor einem Usability-Test ist nach wie vor das Gesamtsystem hochzufahren und zu überprüfen. Es muss gewährleistet sein, dass sämtliche Komponenten einwandfrei funktionieren, da der Test sonst nicht statt finden kann.

   2.1 Zunächst sind sämtliche Geräte, die im Regieraum verfügbar sind, einzuschalten:

   2.1.1 Schnitttrecher
   2.1.2 Capturerechner
   2.1.3 Linkstation
   2.1.4 Monitore des Schnittrechners
   2.1.5 Verstärker für Lautsprecher
   2.1.6 Mikrofonsteuerung
   2.1.7 Metriken – Video – Switch und VGA – Konverter
   2.1.8 VGA – und Metrikenbildschirm
   2.1.9 Kameramonitore

   2.2 Im nächsten Schritt muss der Operator prüfen, ob die Verbindung zwischen den Capturerechnern und dem Laufwerk G:\ vorhanden ist, da die Videoaufzeichnung auf der externen Festplatte gespeichert wird.

2.3 Zusätzlich muss auf beiden Capturerechnern die Software zur Videoaufnahme geöffnet werden (Focus Capture Suite), sonst können die Videokarten der Capturerechner nicht vom Schnittrechner aus angesprochen und die Aufnahme nicht gestartet werden.

2.4 Auf dem Schnittrechner müssen nun Mousecapturer und Buzzertool explizit per Hand gestartet und eingestellt werden.

2.5 Desweiteren ist das Programm zur Videoaufzeichnung zu starten.

2.6 Im Testraum ist der Testrechner hochzufahren. Momentan sind hier zwei Rechner verfügbar, wobei der eine Zugang zu dem Intranet der HAW Hamburg besitzt, so dass auch zu testende Programme direkt aus den Public-Verzeichnissen der Studenten gestartet werden können und nicht extra auf dem Rechner installiert werden müssen. Dies ist insbesondere für Praktika von Lehrveranstaltungen wie Software Engineering von Vorteil.
Es ist also vom Operator zu entscheiden, welcher der beiden Rechner für die vorzunehmenden Tests zum Einsatz kommen wird.

2.7 Auf dem Testrechner werden die Programme von Mousecapturer und Buzzertool automatisch gestartet. Der Operator muss nur noch den Mousecapturer aus dem Blickfeld des Probanden ziehen.

2.8 Desweiteren müssen nun alle 6 Kameras eingeschaltet werden. Die Funktionalität dieser Kameras kann überprüft werden, indem sichergestellt wird, dass auf den 6 Kameramonitoren im Regieraum die einzelnen Live – Bilder der Kameras zu sehen sind.

2.9 Nun muss noch der Eyetracker sowie die zugehörige Software Tobii Studio auf dem Testrechner gestartet werden [Richter 2008].

2.10. Für die Aufzeichnung ist ein synchroner Start aller dafür benötigter Prozesse notwendig, d.h. der Klick auf die vorgesehenen Buttons für den Start der Aufzeichnungen in der jeweiligen Software muss unmittelbar hintereinander erfolgen: Start der Filmaufnahme über den ersten Capturerechner, Start der Filmaufnahme über den zweiten Capturerechner, Start der Aufzeichnung des Eyetrackers, Start der Aufzeichnung der Mausmetriken.

Genaueres zum Hochfahren des Gesamtsystems kann im Anhang B (Bedienungsanleitungen) nachgelesen werden.

3. Starten und Beenden der Aufzeichnung

Gibt der Testleiter das Signal, dass das Vorgespräch abgeschlossen wurde und mit dem Test begonnen werden kann, so wird die Video – und Tonaufzeichnung mit der entsprechenden Software vom Schnittrechner aus gestartet. Nach Ende des Tests müssen die Aufnahmen ebenso wieder angehalten werden. Da es schon zu Fällen gekommen ist, in denen die Capturerechner trotz Stoppen der Filmaufnahme über dem Schnittrechner die

Aufzeichnung der Kameras fortgesetzt haben (vgl. Kapitel 5.1), muss der Operator dies unmittelbar im Anschluss direkt auf den beiden Capturerechnern nachprüfen.

4. Konvertierung der Rohfilme

   VirtualDub soll auch weiterhin als Schnittprogramm eingesetzt werden, da es zum einen skriptbasiert läuft und zum anderen keine grosse Einarbeitung notwendig macht wie andere Schnittprogramme (Edius Pro, Premiere und andere). Nun hat VirtualDub das Problem, dass es nicht vernünftig mit MPEG-Dateien umgehen kann. Aus diesem Grund ist es erforderlich, die einzelnen Rohfilme, die im MPEG – Format aufgenommen werden, nach AVI umzuwandeln. Für diesen Workflow existiert ein Converter als Verknüpfung auf dem Desktop des Schnittrechners, der unter anderem auch dazu benutzt werden kann, den Filmclip des Eyetrackers an die anderen Rohfilme anzupassen (Im Gegensatz zu den Kameras, die mit einer Framerate von 25fps aufnehmen, liegt die Aufzeichnung des Eyetrackers bei 15 fps. Wird die Framerate des Eyetrackers, sofern dieser zum Einsatz kommt, nicht auf 25 fps angehoben, treten bei der Filmzusammenstellung unerwünschte Effekte wie unterschiedliche Geschwindigkeiten auf).

5. Filmschnitt

   Es kann sein, dass die einzelnen Rohfilme nicht vollständig synchron zueinander sind. Dies liegt vor allem daran, dass die Kameraaufzeichnungen herstellerbedingt nacheinander mit einer geringen, dennoch spürbaren Verzögerung gestartet und gestoppt werden. Dies macht derzeit eine Nachbearbeitung der einzelnen Filme durch Zurechtschneiden nötig.

6. Zusammensetzung des Videoclips

   Im nächsten Schritt wird aus den einzelnen Teilfilmen ein einziger Videoclip erzeugt. Hierfür kann VirtualDub benutzt werden. Dieser Arbeitsschritt hat sich im Vergleich zum alten System nicht geändert, soll jedoch weitestgehend automatisiert werden, um den Operator zu entlasten.

7. Sicherung

   Anschliessend müssen die fertigen Filme archiviert und gegen unbeabsichtigte Löschung gesichert werden.

8. Bereitstellung der Filme
   Die fertigen Filme sollten dann vom Operator zur Auswertung im Public – Verzeichnis bereitgestellt werden.

9. System herunterfahren

   Zum Abschluss müssen alle verwendeten Rechner (Capturerechner, Testrechner, Schnittrechner) heruntergefahren und sämtliche Geräte ausgeschaltet werden. Die Reihenfolge spielt dabei keine Rolle.

2.4 Vision für die Zukunft

Im Gegensatz zum Vorgängermodell sind der Aufnahmevorgang und der Herstellungsprozess von Videos deutlich verbessert worden. Filme liegen nun dem Auswertenden schneller vor als früher. Auch der Arbeitsaufwand konnte etwas verringert werden. Jedoch gibt es noch einiges zu verbessern, das langfristige Ziel eines optimal nutzbaren Labors ist noch längst nicht erreicht. Die Möglichkeiten zur weiteren Verbesserung des Systems scheinen unbegrenzt.

Die Vision besagt, möglichst viele Arbeitsschritte zu automatisieren sowie die Anzahl der Arbeitsschritte weiter zu verringern. Geht man beispielsweise davon aus, dass alle Kameras synchron starten, so würde das Schneiden der Filme gänzlich wegfallen. Dieses sollte auf jeden Fall ein weiterer Schritt in naher Zukunft sein.

Ausserdem besteht der Wunsch, sämtliche Tools, die bislang einzeln gestartet werden müssen, über eine einzige Oberfläche steuern zu können. Der Traum wäre es, eine Oberfläche zu haben, mit der gleichzeitig Mousecapturer und Kameraaufzeichnung bedient werden können. Operator und Testleiter sollen möglichst wenig manuell machen müssen, im Idealfall sogar gar nichts.
Der Operator sollte lediglich die Aufnahme starten (START-Button), stoppen (STOP-Button) sowie die vollständig automatisierte Nachbereitung per Mausklick anwerfen (BYE).



Abb. 2-4-1: Wunschvorstellung einer idealen Benutzeroberfläche

Die Implementierung einer solchen Oberfläche ist aufgrund der technischen Gegebenheiten derzeit jedoch noch nicht möglich. Allerdings kommt das entwickelte Operatingtool (vgl. unter anderem Kapitel 3.8) der Vorstellung des BYE-Buttons schon ziemlich nahe.

Eine weitere Wunschvorstellung ist es, den fertig gerenderten Film schon während der Aufnahme zur Ansicht parat zu haben, spätestens jedoch kurz nach Durchführung des Tests. Dies würde eine Testauswertung gemeinsam mit dem Probanden direkt nach der Durchführung erlauben. Im Augenblick dauert das Zusammenschneiden des Films, abhängig von der Länge des Tests, doch noch eine ganze Weile.

Ferner besteht in Einzelfällen der Wunsch, Filme nicht mehr im AVI – Format, sondern im neueren WMV - Format zu erhalten. Dies kann schon jetzt realisiert werden, denn der in Kapitel 2.2 angesprochene Konverter bietet die Möglichkeit, Filme auch in das WMV – Format zu konvertieren. Allerdings ist dazu ein zusätzlicher

Arbeitsschritt nötig (Konvertierung des fertigen AVI-Films nach WMV), was sicherlich noch optimiert werden könnte.

Für den Testleiter könnten noch weitere Auswertungstools geschaffen werden. Ein Beispiel wäre hier eine automatisierte Reporterstellung, die es dem Testleiter möglich macht, schon während des Tests mit der Auswertung zu beginnen. Dies würde den Vorteil erbringen, dass nur wenige Details und Aspekte, die bei dem Test auftreten, später verloren gehen. Denn eine mehrmalige Ansicht der Filme im Nachhinein kann niemals die Eindrücke wiedergeben, die „live" während des Usability-Tests in Erscheinung treten.

2.5 Anforderungsplanung

Damit die in Kapitel 2.2 dargestellten Workflows möglichst einfach gehalten werden können und von dem Endanwender ohne grossen Mehraufwand bewältigt werden können, ist es notwendig, die bereits vorhandenen Tools des alten Labors auf die neuen Gegebenheiten umzustellen, oder aber diese neu zu entwickeln.
Endanwender können hierbei unterschiedliche Personen mit unterschiedlichen Funktionen sein, wie beispielsweise der Testleiter, der spezielle Auswertungstools verwendet, um nach den Testsitzungen leichter zu einem Gesamtergebnis zu kommen, oder aber der Operator, der die Technik des Labors dazu ausnutzt, um dem Testleiter beispielsweise die benötigten Materialien für dessen Auswertung aufzubereiten.

Die folgende Auflistung ist in zwei Abschnitte aufgeteilt. Im ersten Teil sind die Anforderungen genannt, die sich aus den Workflows des Operators ergeben.
Im Anschluss daran sind Anforderungen genannt, die sich nicht unmittelbar auf die Workflows beziehen, sondern dessen Umsetzungen in konkrete Hilfswerkzeuge eine Erleichterung für den Testleiter darstellen sollen.

In beiden Teilen werden Anforderungen selbst nochmal in die Bereiche „Must",
welche Anforderungen darstellt, die als zwingend gelten und auf die nicht verzichtet werden kann, sowie „Should", worunter ebenfalls notwendige, aber nicht zwingend notwendige Anforderungen zu verstehen sind, und „Nice to have" eingeteilt. Letzteres sind Anforderungen, die zwar wünschenswert sind, aber nur zu komfortablen Zwecken dienen und somit am ehesten auf einen späteren Zeitpunkt verschoben werden können.

Zu den Anforderungen, die sich aus den Workflows des Operators ergeben, gehören:

- *Rendern:*
  Während eines Tests laufen viele Datenströme in Form von Videosignalen sowie einer Tonspur auf den beiden Capturerechnern ein. Im äussersten Grenzfall erhält man somit bis zu 7 einzelne Rohfilmdateien und einer weiteren, in der auch die Audiosignale festgehalten sind (Zuletzt waren diese in dem Film enthalten, der von der Kamera 1 aufgezeichnet wurde). Nun kann es einem Testleiter nicht zugemutet werden, alle Rohfilme zu einem einzigen Test einzeln betrachten zu müssen. Eine Auswertung ist unter diesen Voraussetzungen nahezu unmöglich, da davon ausgegangen werden kann,

dass man dabei alle Filme nicht nur ein einziges Mal durchzugehen braucht, sondern dass Filme mehrmals angeschaut werden müssen. Würde eine Testaufzeichnung nun 20 Minuten dauern, und hätte der Testleiter sich alle Filme der einzelnen Kameras nacheinander anzuschauen, so ergäbe sich für die einzelne Durchsicht eines einzigen Films schon eine Gesamtzeit von 8 x 20 Minuten, also nicht ganz drei Stunden.

Ein anderer Aspekt ist der, dass Auffälligkeiten besser erkennbar sind, wenn alle Kamerasichten in einem einzigen Film gleichzeitig angesehen werden können. Andererseits wäre es denkbar, dass sonst die auswertende Person auf bestimmte oder wichtige Details zu spät aufmerksam wird oder diese sogar vollständig übersieht.
Aus diesem Grunde ist es notwendig, dass alle entstandenen Rohdaten in eine akzeptable Bildqualität gerendert und zu einem einzigen Film zusammengeschnitten werden.

Diese Anforderung stellt ein absolutes Muss dar (Must).

- *Sichern und Archivieren:*
  Da auch einige Partnerfirmen im Rahmen des Usability-Engineerings das Labor nutzen, kann jederzeit der Fall eintreten, dass beispielsweise ein weiteres Ansehen oder eine erneute Kopie des Films gewünscht wird.
  Auch aus rechtlichen Gründen (Urheberrecht) kann eine Archivierung vorteilhaft sein. Dies sind zwei von vielen Gründen, weshalb eine Archivierung des Films durchgeführt werden sollte.
  Wie auch immer, es muss möglich sein, auch zu späteren Zeitpunkten auf einen älteren Film zurückgreifen zu können.

  Nun bringt eine einfache Speicherung der Filme keine vollständige Garantie zum Erhalt der Filme mit sich. Zum einen besteht kein Schutz vor einer unbeabsichtigten Löschung des Films durch eine Person, die Zugang zu dem Schnittrechner besitzt. Zum zweiten gibt es immer ein gewisses Risiko, dass auch technische Fehler auftreten können. Als Beispiele seien hier Festplattendefekte oder Systemausfälle genannt.

  Ein weiterer Grund ist der, dass im Laufe der Zeit durch immer mehr Filme die Speicherkapazität des Rechners und somit gleichzeitig die Rechenleistung unnötig verringert werden würde.

  Daher ist es unabdingbar, fertig gerenderte Filme an einem anderen, zusätzlichen Ort zu sichern und aufzubewahren. Da dies weitestgehend automatisiert geschehen soll, ist für diesen kleinen, aber bedeutsamen Schritt ebenfalls ein kleines Tool zu entwickeln, das die Filmarchivierung vornimmt.

  Diese Anforderung stellt ein absolutes Muss dar (Must).

- *Automatisches transferieren von fertigen Filmen ins Public-Verzeichnis:*
  Die fertigen Filme müssen dem Testleiter anschliessend auf irgendeine Art und Weise zur Auswertung ausgehändigt werden können. Da das Rendern einige Zeit in Anspruch nimmt und man nicht davon ausgehen kann, dass der Testleiter zu jeder Zeit persönlich erreichbar ist, muss ein Weg geschaffen

werden, wie ein Film den beteiligten Personen zugänglich gemacht werden kann. Da Filme in einem höheren Grade der Vertraulichkeit unterliegen, sollten sie nicht per E-Mail versendet werden. Darüberhinaus können Filme – abhängig von Kameraanzahl und Testlänge – unter Umständen recht gross werden (20 Minuten entsprechen derzeit in etwa 85 MB Speicherplatzbedarf), was Schwierigkeiten beim Transport auf elektronischem Wege mit sich bringen könnte.

Eine Möglichkeit ist es, fertige Filme auf CD / DVD zu brennen. Dies wird nach jetzigem Stand auch so gehandhabt. Nachteil ist, dass sich der Testleiter die Filme bei Gelegenheit abholen müsste. Desweiteren entsteht ein zeitlicher Mehrauswand für die Erstellung der DVD, der vermieden werden kann.

Im Zuge der Automatisierung kommt jedoch eine zweite Möglichkeit in Betracht. Dazu ist es notwendig, für das Usability-Labor ein eigenes Public-Verzeichnis innerhalb des Netzwerkes der HAW Hamburg anzulegen. Hier könnten Kopien der einzelnen Filme automatisch abgelegt werden, so dass der Testleiter selbst zu jedem erdenklichen Zeitpunkt sich seinen Film aus diesem Verzeichnis abholen könnte. Ausserdem würden auch allen anderen beteiligten Personen auf diese Weise Zugang zu den Filmen verschafft, ohne möglicherweise viele Kopien der Filme anfertigen zu müssen. Bislang gibt es dieses Public – Verzeichnis jedoch noch nicht, da hier noch einige Überlegungen und Sicherheitsvorkehrungen zu treffen sind. Zum einen unterliegen die fertigen Filme der Vertraulichkeit, demnach dürfen nur bestimmte Personen von aussen Zugriff auf die Filme haben. Deshalb müssen Unterverzeichnisse im Public – Verzeichnis eingerichtet werden, die passwortgeschützt sind. Dies ist zwar möglich, wurde jedoch verschoben und ist daher nicht Gegenstand dieser Arbeit.

Diese Anforderung dient nur dem Komfort und ist daher zwar wünschenswert, aber nicht unbedingt notwendig (Nice to Have).

- *Operatingtool:*
  In den oberen drei Anforderungen sind die hauptsächlichen Workflows des Operators wiedergegeben. Diese Schritte erfordern viel Zeitaufwand für den Operator, wenn er sie selbst nach jedem Testtag sequentiell ausführen müsste. Alternativ könnten diese Einzelschritte zu einem einzigen zusammengefasst und automatisiert werden. Der Operator müsste nur noch den Vorgang starten, was für ihn eine Arbeitserleichterung wäre.

  In Abbildung 2-3-1 wurde die bestmöglichste Benutzeroberfläche skizziert, durch die zukünftig das Labor optimal gesteuert werden könnte. Derzeit ist dies jedoch nur eine Wunschvorstellung, eine Vision, die technisch noch nicht umsetzbar ist.
  Mit der Realisierung des Operatingtools kann aber diesem Traum ein Stückchen näher gekommen werden.

  Diese Anforderung stellt ein absolutes Muss dar (Must).

- *Ersetzung des VGA – Films durch einen Screenshot – Strip*:
  Durch die Umwandlung des VGA – Bildes in ein analoges Signal, was beim Speichern des zugehörigen Teilfilms passiert, geht die hohe Bildauflösung verloren. Der VGA – Clip wird an die übrigen Kamerafilme angepasst. Dadurch erscheint das Bild des VGA – Schirmes auf dem zusammengefügten Clip matter und undeutlicher. Es lässt sich nicht mehr so viel erkennen wie auf dem ursprünglichen Abbild, so wie es im Regieraum angeschlossenen VGA – Monitor während des Live – Tests der Fall ist. Informationen für den Auswertenden könnten bei der Reproduktion des Testes im schlimmsten Fall verloren gehen.

  Eine Änderung der VGA – Aufzeichnung wäre daher von Vorteil für den Testleiter. Vorstellbar wäre es, den VGA – Film durch eine Serie von Screenshots zu ersetzen und somit das ursprüngliche Abbild zu erhalten.

  Diese Anforderung dient jedoch nur dem Komfort und ist daher zwar wünschenswert, aber nicht unbedingt notwendig (Nice to Have).

Zu den Anforderungen, die sich aus den Workflows des Testleiters ergeben, und die zu einer Erleichterung für den Testleiter bei der Testdurchführung und -auswertung führen sollen, gehören:

- *Mousetracer:*
  Der Mousetracer soll im Zusammenspiel mit dem Mousetracker ein weiteres Auswertungstool darstellen, mit dem der Testleiter die einzelnen Mausbewegungen nachverfolgen kann. Es soll ihm dabei visuell dargestellt werden, wo die Testperson sich zu welchen Zeitpunkten mit der Maus aufgehalten hat. Unter anderem soll dadurch ermittelbar werden, wie lange der Benutzer an einem Punkt mit der Maus ausgeharrt hat.
  Dies würde beispielsweise darauf hindeuten, dass dieser Punkt ein bestimmtes Interesse geweckt hat, woraus der Testleiter weitere Rückschlüsse ziehen kann, abhängig von der Art der Anwendung.

  Der vorhandene Mousetracker ist ein Metrikentool, das die Anzahl von Klicks mit der rechten, linken und mittleren Maustaste sowie die Tastaturanschläge und Bewegungen des Mausrades protokolliert. Die aktuelle Position der Maus liegt in Bildschirm – Koordinaten vor. Ausserdem wird die insgesamt mit der Maus zurückgelegte Strecke gemessen. Diese Daten werden halbsekündlich abgefragt und als XML – Protokoll weggespeichert. [Ulas 2005]

  Bislang gab es nur die Möglichkeit, einfach Diagramme aus den gespeicherten Werten zu erzeugen. Diese Diagramme können zwar zur Veranschaulichung der gewonnenen Daten und zu Vergleichszwecken zwischen den Probanden untereinander verwendet werden, liefern aber keine tatsächlichen Informationen darüber, welche Intuitionen die Testpersonen mit den jeweiligen Mausaktivitäten beabsichtigten. Der Mousetracer soll genau das leisten.

  Inhaltlich werden an den Mousetracer weitere Anforderungen gestellt, die im Kapitel 3.1 aufgelistet sind.

  Diese Anforderung stellt ein absolutes Muss dar (Must).

- *Buzzertool:*
  Das Buzzertool stellt eine erhebliche Erleichterung für den Testleiter bei der Testdurchführung dar, da es ihm die Arbeit nahezu vollständig abnehmen, sich Notizen während des Tests zu machen. Der Testleiter soll sich stattdessen voll und ganz auf den eigentlichen Test konzentrieren können.

  Das Buzzertool ist eine Metallbox, die mit zwei grossen sogenannten „Buzzern“, wie man sie in etwa aus Quizsendungen kennt, sowie vier kleineren Druckknöpfen in den Farben rot, grün, schwarz und weiss ausgestattet ist. Zusätzlich ist ein Mikrofon fest eingebaut.
  Ein Foto des Buzzertools ist in der Abbildung 1 – 2 – 6 auf Seite 17 zu sehen.
  Angeschlossen wird das Buzzertool an den Schnittrechner im Testleiterraum und kann während einer Untersuchung vom Testleiter ohne grosse Mühe bedient werden. Es ist somit eine ideale Unterstützung für den Testleiter.
  Zusammengebaut wurde das Buzzertool bereits in einem früheren Projekt im Rahmen des Projektpraktikums Usability - Engineering. [Zlotopolski 2006]
  In dieser Arbeit soll die dazu vorhandene Software verbessert und erweitert werden.

  Einerseits sollen mit ihm Sprachaufzeichnungen getätigt werden, die die bisherige umgangssprachliche Protokollierung auf Papier ersetzen soll. Andererseits können mit ihm Screenshots vom Testrechner gemacht werden, die dem Testleiter zusätzliche Informationen einfacher und schneller liefern soll, ohne dass er nach den jeweiligen Stellen im fertigen Film lange suchen muss. Screenshots werden separat als Bild abgespeichert. Der Testleiter hat somit einen sofortigen Zugriff auf diese Screenshots.
  Die Implementierung dieser beiden Funktionalitäten ist bereits vorhanden, muss im Zuge des Umbaus jedoch an die neuen Gegebenheiten angepasst werden. Desweiteren wird ein verbesserter Abspeicherungsmechanismus der entstehenden Bild- und Tondateien eingeführt, um zusammengehörende Dateien auch als solche erkennbar zu sichern.

  Um statistische Aussagen über mehrere Tests hinweg zu treffen, wird als neue Funktionalität ein Aufgabenzähler implementiert und dessen Ergebnis als einfache Textdatei mit abgespeichert. Anhand dieser Daten soll dem Testleiter ein Vergleich möglich sein, welcher Zeitaufwand bei den einzelnen Aufgaben vorhanden war, ob Testpersonen unterschiedlich lange für ein und die selbe Aufgabe gebraucht haben, ob Aufgaben komplett nicht gelöst werden konnten und vieles mehr. Auch aus diesen Informationen lassen sich Rückschlüsse ziehen, bei welchen Aufgaben es beispielsweise die meisten Schwierigkeiten gab.

  Diese Anforderung stellt ein absolutes Muss dar (Must).

- *Tool zur Reporterstellung und Auswertung:*
  Nach Ablauf der Tests und des Zusammenschneidens der Filme beschäftigt sich der Testleiter mit der Auswertung. Hierbei wird er mit einer Vielzahl von Daten konfrontiert, die er begutachten muss, und aus denen er Informationen und Rückschlüsse zu ziehen hat, um darauf Empfehlungen aussprechen zu können. Üblicherweise bekommt er nicht nur einen Film zur Auswertung,

sondern gleich mehrere auf einmal. Zu erwähnen ist auch, dass es nicht ausreicht, einen Film nur einmal anzuschauen. Der Auswerter muss sich die Filme immer und immer wieder anschauen, um ein aussagekräftiges Gutachten erstellen zu können. Hinzu kommen noch diverse andere Daten wie die des Mousecapturers, die zu inspizieren sind. Somit ist die Auswertung eine sehr zeitintensive Angelegenheit.

Die ersten Ergebnisse werden möglicherweise schon während des Tests deutlich. Nur hat der Testleiter kaum die Möglichkeit, die Zeit und die Konzentration, um diese festzuhalten. Desweiteren sind manche Daten von ihrer Art her in allen Testdurchgängen identisch oder ähneln sich.

Schön wäre es deshalb, den Testleiter bei der Auswertung so zu unterstützen, dass ihm die meiste Arbeit schon von vorn herein abgenommen wird. Prägnante Daten, die während eines Tests gesammelt werden (u.a. vom Mousecapturer), könnten schon im Vorfeld automatisch ausgewertet
und in einem Report dokumentiert werden. Dies würde eine effizientere Auswertung ermöglichen.

Diese Anforderung dient jedoch nur dem Komfort und ist daher zwar wünschenswert, aber nicht unbedingt notwendig (Nice to Have).

- *Flexiblere Anzeige von fertigen Filmen:*
  Bisher existiert nur eine feste Darstellung der Filme, wobei der VGA – Bildschirm den Hauptbestandteil bildet. Die anderen Kameraperspektiven sind um das VGA – Bild herum platziert. Dies ist insofern ungünstig, da der Betrachter seinen Blick auf diesen Hauptteil fixieren wird, er kann nicht alle Perspektiven gleichzeitig im Auge behalten. Besonders für die beiden neuen Kameraperspektiven ist diese Darstellung eher nicht optimal. Denn gerade diese Sichtweisen dürften für den Auswertenden nicht uninteressant sein, da sie den Probanden direkt von vorne aufzeichnen und sich somit einiges aus dessen Gesichtszügen ablesen lassen könnte.

  Besser wäre eine Darstellung, die den Auswertenden die Möglichkeit gibt, die Perspektive selbst auszuwählen, die gerade im grossen Format angezeigt werden soll, und zwischen den einzelnen Perspektiven während des Abspielens des Clips umschalten zu können. Wenn auf einer der anderen Ansichten etwas Interessantes passiert, könnte die zugehörige Anzeige leicht vergrössert werden und Details somit leichter erkenntlicher machen. Das Rendern der Filme sollte im 16:9 – Fernsehformat geschehen. Optimal wäre eine Bildschirmauflösung von 1280 x 1024 Pixeln.

  Diese Anforderung dient jedoch nur dem Komfort und ist daher zwar wünschenswert, aber nicht unbedingt notwendig (Nice to Have).

Kapitel 3 – Neukonzeption des Labors

Ziel dieser Arbeit soll es sein, die Workflows im Usability-Labor zu verbessern und zu erleichtern. In dieser Hinsicht gibt es viele Einzelziele, zu denen Konzepte formuliert werden. Die Ziele wurden nach Prioritäten verfolgt (vgl. Kapitel 2.4) und deren Realisierung im Rahmen der verfügbaren Kapazität durchgeführt. Dieses Kapitel beschreibt, welche einzelnen Ideen und Konzepte hinter den konkreten Einzelzielen stehen.

3.1 Konzept des Mousetracers

Der Mousetracer stellt ein weiteres Tool zur Auswertung von Filmen dar.
Es nimmt die vom vorhandenen Mousecapturer zur Verfügung stehenden
XML-Daten (zu denen gehören u.a. Anzahl Mausklicks auf den drei Mausbuttons sowie Anzahl Aktivitäten mit dem Mausrad) als Eingabe entgegen. Insbesondere spielt hierbei die Mausposition im Verlauf eines Testfilms eine Rolle.

Als Vorbild darf der neu beschaffte Eyetracker gelten, der ähnliches für die Blickverfolgung leistet. Der Eyetracker tastet per Infrarot die Augen der Testperson ab und stellt die Blickposition, also den Punkt, auf den die Testperson gerade zu der Zeit seine Aufmerksamkeit richtet, in Form eines roten Punktes grafisch dar.
Dieses wird für den gesamten Testverlauf aufgezeichnet und ist später als eigener Filmclip verfügbar und kann ausgewertet werden.
Dieses soll nun auch für die Mausbewegungen realisiert werden. Die Positionen der Maus werden hier allerdings nicht während des Tests neu aufgezeichnet, sondern werden im Anschluss aus den angesprochenen XML – Daten ausgelesen und weiterverarbeitet. Die Mauspositionen sollen dann ebenfalls nachgestellt und als grafische Punkte dargestellt werden. Verbunden durch eine Linie werden die Positionsveränderungen angezeigt, jeweils in einem geeigneten Zeitabschnitt.
Daraus entsteht ein weiterer Clip, der für die Auswertung herangezogen werden kann. Über die Mauspositionen kann so herausgefunden werden, welche Bereiche des Desktop – Bildes für den Benutzer als interessant empfunden wurden.
Bleibt der Mauscursor auf einem Punkt länger stehen als auf anderen, so kann daraus ersehen werden, dass an dieser Position ein bestimmtes Merkmal oder eine Funktion der Anwendung untergebracht ist, die den Probanden längere Zeit beschäftigte.
Weiter ist es möglich, bestimmte Bereiche zu lokalisieren, in denen die Maus ganz häufig bewegt wurde, genau wie solche Bereiche, in denen die Maus nie anzutreffen war. Somit lässt sich beispielsweise für Webseitenuntersuchungen nachweisen, dass bestimmte wichtige Funktionen wie die Suche nicht verwendet werden, wenn sie schlecht auf der Webseite platziert sind oder schwer erkennbar sind. In solchen

Fällen wird dann die Maus nicht in die Nähe des eigentlich interessanten Objekts bewegt, was sich mit dem Mousetracer gut beweisen lassen kann. [Nelius 2003]

Eine Darstellung in Form einer Heatmap könnte eine solche Funktionalität am geeignetsten visualisieren.

Auch für andere Untersuchungen im Hinblick auf die Verwendung der Maus könnte der Mousetracer ein geeignetes Instrument sein, bestimmte Vermutungen nachzuweisen und Verhaltensmuster aufzudecken. So könnte man zeigen, dass es Menschen gibt, die beim Lesen eines Textes die Maus gemäss der Leserichtung mitführen oder als Orientierungshilfe nutzen.

Aufgrund der Ähnlichkeit mit dem Eyetracker können die aufgezeichneten Bilder und Videos des Mousetracers mit denen des Eyetrackers kombiniert werden. Somit hätte man beide Verläufe (Blickverlauf und Mausbewegungen) auf einem Blick. Das wiederum könnte neue Möglichkeiten für Untersuchungen eröffnen, die sich mit der Frage nach dem Zusammenhang und des Zusammenspiels von Maus und Blick beschäftigen. Ein solches Beispiel wäre die Untersuchung der Hypothese, dass die Maus immer dem Blick des Benutzers folgt. Punkte auf dem Bildschirm, die mit der Maus aufgesucht werden, wurden vorher angesehen (Hand – Auge – Koordination).
Diese Hypothese kann aber nur zum Nachweis der Funktionsfähigkeit und zum Sammeln von Erfahrungen dienen. Die bekannten Zusammenhänge zwischen Auge und Hand sollten in einem ersten Schritt von der neuen Technologie bestätigt werden (vgl. hierzu [Richter 2008]).

Der Mousetracer hat also prinzipiell dieselben Aufgaben und Eigenschaften wie der Eyetracker. Für die konkrete Realisierung des Tools werden folgende Anforderungen erkannt:

- 1. Import der Basisdaten

  Zunächst einmal muss eine Möglichkeit geschaffen werden, die zugrunde liegenden Daten in den Mousetracer laden zu können. Da der Mousetracer keinen Live – Mitschnitt bieten soll, müssen die benötigten Daten auf andere Weise beschafft werden, um sie gemäss den Anforderungen weiterzuverarbeiten.
  Die Daten werden vom Mousecapturer in Form einer XML – Datei geliefert.
  Es ist daher ferner zu beachten, dass die zu entwickelnde Schnittstelle den Import solcher Dateien unterstützt.

  Diese Anforderung stellt ein absolutes Muss dar (Must).

- 2. Extrahieren der tatsächlich benötigten Daten

  Der Mousecapturer liefert eine Menge an Informationen, die in dieser XML – Datei abgespeichert sind. Diese sind Tastaturanschläge, Mauspositionen in X – sowie in Y – Richtung, Mausklicks mit linker, mittlerer und rechter Maustaste, Mausstrecke in Metern und zu guter letzt Mausradaktivitäten.
  Sie werden jeweils in halbsekündlichen Abständen als einzelne XML – Knoten protokolliert angegeben.
  Tatsächlich benötigt werden aber nur die Daten bezüglich der Positionen.
  Diese Angaben müssen daher mit einem geeigneten XML – Parser aus der Datei extrahiert werden.

  Diese Anforderung stellt ein absolutes Muss dar (Must).

- 3. Visualisierung der Mauspositionen

  Die Mauspositionen müssen in geeigneter Weise wiedergegeben werden.
  Die beste Möglichkeit ist die Darstellung als Kreis. Die Darstellung selbst darf nicht zu klein gewählt werden, es ist also ein geeigneter Durchmesser für die Zeichnung zu ermitteln, so dass der Kreis gut sichtbar ist. Die Farbe spielt dabei auch eine wichtige Rolle. Sie darf nicht zu hell und nicht zu dunkel gewählt werden, so dass die Zeichnung auf jedem Hintergrundbild zu erkennen ist. Rot scheidet dabei aus, da die generierten Endprodukte später mit denen des Eyetrackers zusammengefügt werden sollen und die Blickverläufe vom Eyetracker in roter Farbe angezeigt werden. Die Farbe sollte frei einstellbar sein.

  Diese Anforderung stellt ein absolutes Muss dar (Must).

- 4. Bildschirmauflösung für die erzeugten Bilder

  Die Darstellung der Positionen muss exakt den Angaben in den XML – Dateien entsprechen. Deshalb ist darauf zu achten, dass die zu erzeugenden Einzelbilder genau derselben Auflösung unterliegen, wie sie auf dem Testrechner gegeben ist.

  Diese Anforderung stellt ein absolutes Muss dar (Must).

- 5. Aneinanderreihung der Bilder als „Video“

  In halbsekündlichen Abständen werden neue Daten aus der XML – Datei gelesen. Die Darstellung der Mauspositionen ist daher Änderungen gegenüber anzupassen. In diesem Sinne muss für jede eingelesene Positionsangabe ein neues Bild erzeugt werden.

Um eine einfache Handhabung des Tools zu garantieren, ist ein automatisches Abspielen der Einzelbilder entsprechend der gegebenen Reihenfolge erwünscht. Dies sollte in Form eines Videoclips möglich sein.

Die Positionen der Maus sollten bei Bildwechsel so angezeigt werden, dass jeweils die neue Position und die alte Position als Kreis dargestellt, und beide Kreise durch eine Linie verbunden sind. Ändert sich die Position nicht, so soll der Durchmesser des Kreises vergrössert werden, um ein Innehalten der Maus besser zu veranschaulichen.

Diese Anforderung stellt ein absolutes Muss dar (Must).

- 6. Wiedergabe der Bilder als Video

  Für die Wiedergabe des entstehenden Films ist eine Komponente zu entwickeln oder eine bestehende einzubinden, die es ermöglicht, die Einzelbilder als Ganzes abzuspielen. Diese Komponente sollte die üblichen Funktionen wie Start, Stop, Vor – und Zurückspulen sowie Pausieren des Films besitzen. Die Video – Dateien sind gegenwärtig mit dem Video – Kompressor XviD codiert. [XviD 2006]
  Daher muss die Komponente auf diese Weise codierte Filme unterstützen können und eine angemessene Fehlerbehandlung bieten.

  Diese Anforderung stellt ein absolutes Muss dar (Must).

- 7. Synchrone Wiedergabe zweier Videofilme

  Um einen Bezug zum Originalbildschirm zu erhalten, wäre es hilfreich, als Hintergrund den Film des VGAs zu verwenden. Durch die Aneinanderreihung der Mousetracer – Bilder entsteht jedoch ein zusätzliches Video. Der Videoplayer muss also in der Lage sein, beide Videos synchron abspielen zu können.
  Diese Anforderung stellt ein absolutes Muss dar (Must).

- 8. Transparenz

  Eine zufriedenstellende Auswertung des Videos wird erst dann möglich, wenn die selbst erstellte Darstellung der Mauspositionen in direktem Bezug zum Bildschirminhalt steht. Daher ist es notwendig, beide Bilder, sowohl das der Positionsdarstellung als auch das ursprüngliche VGA – Bild übereinanderzulegen. Dies setzt voraus, dass das Mousetracer – Bild transparent gemacht wird, so dass letztendlich aus beiden Einzelbildern ein einziges entsteht.

  Diese Anforderung stellt ein absolutes Muss dar (Must).

- 9. Heatmaps

  Um die besonders oft mit der Maus angefahrenen Bereiche zu lokalisieren, ist eine andere Darstellung der Mauspositionen nötig. Diese soll nicht die Position in einem zeitlichen Raster darstellen, sondern alle Positionen während einer bestimmten Zeitspanne in einem Bild. Hier bietet sich die Darstellung einer Heatmap an, wobei stark berührte Bereiche dunkler gefärbt sind als weniger stark berührte Bereiche. Die einzelnen Punkte könnten dabei in Gebiete eingruppiert werden, die mit Hilfe von Bezier – Splines gezeichnet werden.
  Diese Bilder sollten ebenfalls dynamisch erzeugt und aneinandergereiht werden, so dass sich die Heatmap Bild für Bild weiter aufbaut und verändert.
  Es ist also eine Schnittstelle zu integrieren, die die Erzeugung von Heatmaps durch den Benutzer unterstützt.

  Diese Anforderung stellt ein absolutes Muss dar (Must).

- 10. Bestimmung des Zeitintervalls für Heatmaps
  Nun bringt die Darstellung der Bereiche über den gesamten Zeitraum, der in dem Videoclip enthalten ist (also das Intervall zwischen Filmstart und Filmende), für den Benutzer nichts. Die Zeichnung wird zu unübersichtlich und es ist kaum etwas zu erkennen.
  Besser wäre es daher, wenn der Benutzer das Zeitintervall selbst bestimmen könnte. Eine Implementierung in Form von Schiebereglern wäre insofern angemessen, da der Benutzer somit gleichzeitig die Möglichkeit erhält, das Intervall zu modifizieren, während die Heatmap dargestellt wird. Somit liesse sich beobachten, wie sich die Heatmap im Laufe der Zeit verändert und somit Verschiebungen des Interessensbereiches der Testperson nachweisen.

  Diese Anforderung stellt ein absolutes Muss dar (Must).

- 11. Export der Heatmaps und Videos in geeignete Formate

  Für die Weiterverwendung ist ein Export der generierten Dateien ein Muss. Dabei sind Formate zu verwenden, die auch von anderen Programmen akzeptiert werden. Heatmaps beispielsweise sollen als Einzelbilder gespeichert werden können, hier bieten sich das Bitmap – Format BMP oder das komprimierte Bildformat JPEG an. Clips sollten als AVI – Datei gesichert werden.

  Diese Anforderung stellt ein absolutes Muss dar (Must).

3.2 Neues Konzept zum skriptgesteuerten Rendern von Rohfilmen

Bisher wurden die Einzelfilme von vier Kameras sowie der VGA-Film und die Aufzeichnung des Mauscapturers direkt im .AVI-Format jeweils auf separaten Festplatten gespeichert. Von dort aus konnten sie mit dem Videobearbeitungs-programm VirtualDub über Avi-Synth-Skripte bequem zu einem einzelnen Film zusammengeschnitten werden.

Durch die Umstellung des Videoaufzeichnungsverfahrens, wobei aktuell bis zu acht Kanäle parallel auf zwei Rechnern aufgenommen werden können (vgl. Kapitel 1.2.2), sowie die Änderung des Aufnahmeformats der Filme (.MPEG) ist auch eine Anpassung des bisherigen Rendervorgangs an die gegebene Situation notwendig geworden. Desweiteren sind zwei weitere Kameras in Betrieb genommen worden, deren Aufnahmen in diesen Arbeitsschritt integriert werden müssen.

Um diesen Workflow möglichst einfach zu halten, soll das Rendern auch zukünftig skriptgesteuert ablaufen können. Die Skripte sollen dabei wie gewohnt vollautomatisch erzeugt werden.

Am Anfang einer Testaufzeichnung wird ferner entschieden, welche Kameras überhaupt zum Einsatz kommen sollen. Nicht immer also werden alle 6 Kameras benötigt. Daraus folgt, dass die Skripterzeugung so flexibel gestaltet werden muss, dass die Anzahl der Kameras möglichst unkompliziert variiert werden kann, ohne eine Beeinträchtigung dieses Arbeitsschrittes mit sich zu führen, und keines persönlichen Eingriffs des Operators per Hand in das Skript bedarf.
Mit anderen Worten: Es muss möglich sein, dass zu jeder möglichen Kombination der Kameras ein passendes Skript generiert werden kann. Da jedoch die Rohfilme der einzelnen Kameras bei der Speicherung der Daten nahezu gleiche Dateinamen verwenden, erscheint dies unproblematisch. Eine Möglichkeit zur Realisierung wäre die Erstellung von vorgefertigten Templates, die flexibel den verschiedenen Aufnahmevarianten entsprechend gefüllt werden.

Nach intensiver Suche und Erprobung von verschiedenen Videobearbeitungsprogrammen stellte sich heraus, dass VirtualDub trotz der Tatsache, dass es hierbei zu Problemen mit dem angesprochenen .MPEG-Format kommen kann (vgl. Kapitel 5.3.2), immer noch das geeignetste Programm zu sein scheint, das die Anforderungen des Usability-Labors am ehesten erfüllt.
Die meisten anderen Videobearbeitungsprogramme haben den grossen Nachteil, dass sie nicht skriptgesteuert bedient werden können. Eine zum Teil recht komplizierte Bedienung per Hand wäre in den meisten Fällen unumgänglich gewesen, was den Prozess der Filmerstellung wesentlich verlängert hätte. Dies ist jedoch nicht das Ziel.

Auch gibt es auf dem Markt derzeit kaum freie oder kostengünstige Produkte,
die es ermöglichen, mehrere, von verschiedenen Kameras erhaltene Rohdaten zu einem Film zusammenzufassen. So bleibt letztendlich VirtualDub als derzeit einzige Alternative übrig.

Demnach muss es also ein Verbindungsglied in Form eines Programmes geben, welches weiterhin die Skripte erzeugt, die im Anschluss mit VirtualDub weiterverarbeitet werden können.
Dieses Programm hat zunächst zu entscheiden, welche Rohfilmdateien jeweils zu einem Test zusammengehören. Desweiteren muss geklärt werden, wie viele Kameras für den einzelnen Test in Gebrauch waren, um entsprechend die einzelnen Kameraperspektiven (Kamera 1 bis 6 sowie VGA) auf dem späteren Endfilm aufteilen zu können und den dafür verfügbaren Platz bestmöglich auszunutzen. Aus diesen Informationen kann dann ein AviSynth-Skript geschrieben werden, das als Eingabe für VirtualDub dient.

Eine weitere Änderung beim Rendern besteht darin, dass aufgrund der Vielzahl an Einzelfilmen und der Hinzunahme zweier weiterer Kameras die Auflösung des Gesamtfilms geändert wurde. Wurden Filme vorher im Format 768 x 576 Pixel (PAL) fertiggestellt, so liegt die Auflösung nun bei etwa 1140 x 880 Pixel. Das hat den Vorteil, dass zum einen auf den einzelnen Abschnitten Details besser erkennbar sind. Zum anderen wurde damit eine Anpassung an die heute üblichen Bildschirmauflösungen (1280 x 1024 Pixel, SXGA oder 1280 x 990 Pixel im 4:3 - Fernsehbildschirmformat) vorgenommen. Ein grosser Nachteil ist allerdings, dass der Rendervorgang selbst etwas zeitaufwändiger ist als vorher, da die Rohdaten nicht in dieser Auflösung vorliegen, sondern dementsprechend erst daraufhin skaliert werden müssen. Diese Skalierung ist Bestandteil des angesprochenen AviSynth-Skriptes.

Abbildung 3 – 2 – 1 zeigt einen Überblick über die heute existierenden Videoformate.
Neben den 4:3 - Videoformaten wie SVGA oder XGA (dargestellt in rot) oder dem
5:4 – Format SXGA (in blauer Farbe) ist dort auch das aktuelle Videoformat der fertigen Filme eingezeichnet, um zu verdeutlichen, welchem Videoformat sich angenähert wurde. Optimal wäre eine Auflösung von 1280 x 960, da das
4:3 – Bildschirmformat beibehalten werden soll.

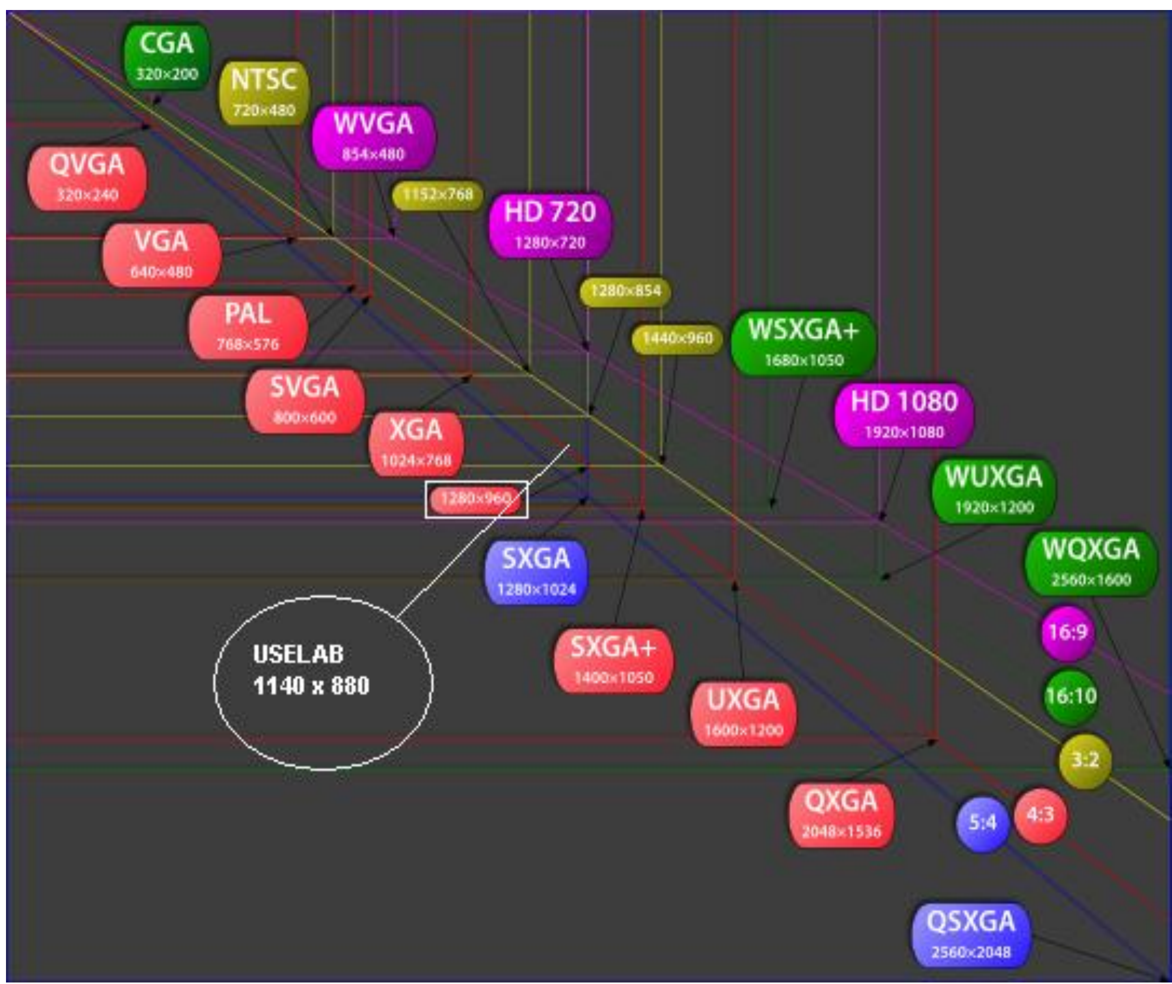


Abb. 3-2-1: Videoformate im Überblick

Die ursprünglich mitgeschnittene Aufzeichnung des Mauscapturers wird nun bei der Erstellung des Endfilms nicht mehr weiter berücksichtigt, kann aber jederzeit als eigenständiger Teilfilm auf Wunsch eingefügt werden. Für die Daten des Mauscapturers stehen andere und bessere Auswertungstools zur Verfügung. Die Anzeige der Mausdaten während des Films würde zu Unübersichtlichkeit führen und den Betrachter eher vom eigentlichen Geschehen, auf das er sich konzentrieren soll, ablenken.

3.3 Konzept zur Sicherung und Archivierung der Filme

Fertig gerenderte Filme müssen gegen unbeabsichtigte Löschung gesichert werden. Dafür steht eine neue Linkstation zur Verfügung, auf der die Filme mittels eines kleinen Programms automatisch gesichert und archiviert werden können. Dieses soll so ablaufen, dass alle noch nicht gesicherten Filme durch einen einzigen Mausklick auf der Linkstation gespeichert werden.

Die Linkstation selbst kann als eigenständige, externe Festplatte angesehen werden. Sie ist von allen Rechnern im Netzwerk aus über das Laufwerk L:\ zugänglich und verfügt über einen Gesamtspeicher von etwa 3 TB. Geht man davon aus, dass ein fertiger Film am Ende beispielsweise eine Speicherkapazität von 75 MB (entspricht in etwa 30 Minuten Film) erfordert, so hätten ungefähr 41.943 Filme auf ihr Platz.
Da nur die Filme von jeweils 3 gleitenden Semestern auf der Linkstation gespeichert werden sollen, eignet sich diese Hardware somit hervorragend als Archiv.
Für die Langzeitarchivierung bieten sich andere Möglichkeiten an. Eine Idee wäre es, interne Festplatten zu verwenden, die temporär angeschlossen werden und auf denen eine weitere Kopie der Filme abgelegt werden (2. Sicherung, s.u.).

Die Archivierung eines Films auf der Linkstation muss dabei vor allem auf einem zuverlässigen, sicheren Weg erfolgen. Dabei kommt es in diesem Schritt nicht so sehr auf eine hohe Geschwindigkeit an. Viel wichtiger ist es, dass die Daten ohne Informationsverlust und unbeschädigt auf die Linkstation transportiert werden, und sich am Ende eine Art „Kopie“ des Originalfilms auf ihr befindet.
Nur so kann gewährleistet werden, dass im Falle eines Systemfehlers oder einer ungewollten Löschung der Originaldatei eine Wiederherstellung aus dem Archiv möglich ist.

Die einfachste Möglichkeit besteht dabei, die einzelnen Bytes des Originals zu kopieren und diese sequentiell über die Netzwerkleitung auf die Linkstation zu übertragen. Das ist sicherlich nicht die schnellste und eleganteste Lösung, jedoch kommt es hierbei selten zu Datenverlusten. Daher reicht diese Art der Übertragung für die Zwecke des Labors völlig aus.

Um eine optimale Sicherheit zu erlangen, sollten – wenn möglich – die einzelnen Dateien immer doppelt kopiert und an getrennten Orten aufbewahrt werden. Vielfach wird die 2.Sicherung gar auf tragbare externe Speichermedien transportiert und beispielsweise in Bankschliessfächern deponiert. In diesem Fall könnte die 2. Sicherung zur Langzeitarchivierung im Serverraum des Softwarelabors untergebracht werden.

Hier kann eine zweite Sicherung am besten im Zuge einer Backup-Sicherung erfolgen, wobei das entsprechende Filmearchiv vom Laufwerk L:\ der Linkstation auf eine Datei abgebildet und komprimiert wird. Selbstverständlich darf sich diese Gesamtsicherungsdatei nicht selbst auf dem Laufwerk L:\ befinden, sondern an einem anderen Ort, damit eine eventuell nötige Wiederherstellung durchgeführt werden kann. Die bestmögliche Alternative für die 2.Sicherung ist wiederum das Laufwerk G:\.

Anders als bei der ersten Sicherung kann dafür das Programm *Acronis True Image* genutzt werden. *Acronis True Image* ist ein kommerzielles Programm, das im Labor zu Recoveryzwecken eingesetzt wird und über das ganze Laufwerke gesichert werden können.

3.4 Automatisches Transferieren der Filme in ein öffentliches Verzeichnis

Ziel ist es hierbei, fertige Filme zusätzlich mit möglichst wenig Aufwand automatisch in ein öffentliches Verzeichnis der HAW Hamburg zu stellen.
Der Vorteil besteht darin, dass später auf die fertigen Filme auf einfachere und elegantere Weise zugegriffen werden kann. Der Auswertende kann sich jederzeit die Filme selbst abholen.

Dies kann unter Umständen parallel mit der Sicherung der fertigen Filme (siehe Kapitel 3.3) geschehen. Dazu wird eine weitere Kopie des Films angelegt, in einen Bytestrom umgewandelt und mit Hilfe eines geeigneten Protokolls über das HAW-Netzwerk in das Public-Verzeichnis transferiert.

Das zuverlässigste Netzwerkprotokoll scheint hierfür FTP (File Transfer Protocol) zu sein, da es sich um eine einfache Folge von Byteströmen handelt, die transportiert werden müsste.
Das FTP-Protokoll ist ein Netzwerkprotokoll zur Dateiübertragung über TCP/IP-Netzwerke und ist in der Anwendungsschicht des OSI-Schichtenmodells angesiedelt. Hauptsächlich können mit diesem Protokoll Dateien zwischen zwei Terminals im Netzwerk (entweder Client-Server oder Server-Client) übertragen werden, aber auch allgemeine Verzeichnisoperationen wie Anlegen eines neuen Verzeichnisses sind mit FTP möglich. Zum Senden und Empfangen von Dateien wird pro Vorgang jeweils eine eigene TCP-Verbindung benötigt. Der Aufbau dieser TCP-Verbindung ist dabei gleichzeitig als Beginn der FTP-Sitzung anzusehen. Über diese Verbindung werden entsprechende Befehle zum Server gesendet, worauf der Server jedesmal mit einem bestimmten Statuscode antwortet. Die meisten Befehle sind allerdings erst nach einer erfolgreichen Authentifizierung zulässig.

Für die Programmierung unter Java bietet Sun eine einfache API an, von der im Wesentlichen zwei Klassen benötigt werden:

- FtpClient, mit der die Verbindung zum HAW-Server auf einfache Art hergestellt werden kann und die den Zugriff auf das Pub-Verzeichnis regelt

- TelnetOutputStream, eine Klasse, mit der Streams durch ein Netzwerk übertragen und am Zielort weggeschrieben werden können

Das Speichern des Films im Pub-Verzeichnis läuft somit prinzipiell ähnlich wie bei der Speicherung auf der Linkstation ab (vgl. hierzu Kapitel 4.3). Lediglich der Ausgabestream zum Schreiben müsste durch einen TelnetOutputStream ersetzt werden.

Die Vorgehensweise könnte wie folgt aussehen:
Mittels der FtpClient-Klasse wird die Verbindung zu dem HAW-Server über die URL „ftp.informatik.haw-hamburg.de“ geöffnet. An dieser Stelle angekommen, wird die Angabe eines Benutzers mitsamt Passwort erforderlich. Dieser Benutzer muss jemand sein, der einen vollständigen Zugriff auf das Public-Verzeichnis hat und vor allem über Schreibrechte verfügt, damit eventuell auftretende Schwierigkeiten mit den Berechtigungen von vornherein vermieden werden können. Über FTP könnte dann in das Publicverzeichnis des Usability-Labors gewechselt werden und dieser Pfad als Zielort der Kopie des Films bestimmt werden. Nach der erfolgreichen Übertragung der Bytefolge wird die Verbindung durch FTP wieder geschlossen.

Nun ist es leider so, dass noch kein Public - Verzeichnis für das Usability-Labor existiert und somit auch noch kein Benutzer mit ausreichenden Berechtigungen angegeben werden kann. Hierfür müsste für jedes Projekt im Public – Verzeichnis ein eigener Ordner erstellt werden, auf dem nur eine bestimmte Personengruppe Zugriff hätte – nämlich die der Auswertenden. Jedoch dürfen die Filme nicht allgemein verfügbar gemacht werden, um Datenschutz gewährleisten zu können.
Daher erweist sich die konkrete Umsetzung dieser Idee zum gegenwärtigen Zeitpunkt noch als schwierig.

Aus diesem Grund konnte das für diese Anforderung zu entwickelnde Programm zum jetzigen Zeitpunkt noch nicht vollständig fertiggestellt werden. Jedoch sind Ansätze und Ideen zur Vorgehensweise (s.o.) bereits vorhanden und können bei Bedarf weiterentwickelt werden.

3.5 BuzzerTool – Ideen und Konzepte

Während einer Untersuchung treten meist Situationen auf, die Probleme der Testpersonen im Umgang mit dem untersuchten Objekt aufweisen. Dies liegt in der Absicht des Testleiters, da viele Aufgaben die Schwachstellen des Objekts aufdecken sollen und so ausgelegt sind, dass Probanden geradezu in diese Schwierigkeiten hineinlaufen. Oftmals findet der Proband selbst einen Weg zur Lösung. Interessant ist jedoch, wie die einzelnen Testpersonen mit den Schwierigkeiten zurechtkommen. Eine gute Hilfe geben der auswertenden Person dabei die zu dem Zeitpunkt vorgenommenen Aktivitäten der Testperson auf dem Bildschirm sowie eventuelle Bemerkungen, die der Proband diesbezüglich macht („Lautes Denken“).

In diesen Fällen macht sich der Testleiter meist Notizen, um die Eindrücke und Schwierigkeiten des Probanden festzuhalten. Da sich gezeigt hat, dass es

üblicherweise nur einen Testleiter gibt, kann dies zu unnötigen Nebeneffekten und Problemen für den Testleiter selbst kommen.
Für die meisten Menschen ist es eine erhebliche Last, sich auf zwei grundsätzlich verschiedene Dinge und Vorgänge gleichzeitig konzentrieren und sich viele Dinge auf einmal merken zu müssen. Durch das Schreiben der gesammelten Informationen auf Papier kann so der Testleiter vom eigentlichen Geschehen abgelenkt werden. Weitere, möglicherweise wertvolle Informationen, die parallel vom Probanden gegeben werden, werden vom Testleiter nicht richtig wahrgenommen oder gehen vollständig verloren.

Um einen solchen Informationsverlust zu vermeiden, wäre es daher wünschenswert, den Testleiter soweit zu entlasten, dass eine Informationssammlung auf Papier weitestgehend reduziert wird. Der Testleiter soll seine vollständige Aufmerksamkeit auf die eigentliche Untersuchung richten können.

Es liegt daher nahe, dass dem Testleiter eine Möglichkeit geschaffen wird, seine Eindrücke über die Sprache aufzuzeichnen anstatt über Papier und Bleistift. Sprechen lenkt weniger ab als Schreiben. Der Testleiter muss so nicht den Blick von den Monitoren abwenden, um Informationen niederzuschreiben. Er könnte vielmehr seine volle Aufmerksamkeit darauf lassen, wenn es ihm möglich wäre, über Mikrofon und Sprachaufzeichnung seine Gedanken zu protokollieren.

Desweiteren wäre es hilfreich, wenn besonders herausragende Situationen und Vorgänge auf dem Bildschirm des Testrechners, die offensichtliche Probleme des Probanden zeigen, direkt und gesondert, d.h. abgekoppelt von der VGA-Aufzeichnung, gespeichert werden könnten. Eine Problemsituation könnte beispielsweise das verzweifelte Navigieren des Probanden durch eine nicht optimal angelegte Verzeichnisstruktur sein.

Problemsituationen werden zwar auf dem VGA-Video mit aufgezeichnet, müssen später aber vom Testleiter auf dem Endfilm erst einmal wiedergefunden werden.
Gibt man allerdings dem Testleiter die Möglichkeit, einen Screenshot vom Bildschirm des Testrechners zu machen, so hätte der Testleiter diese Momente gleich direkt zur Hand.

Hauptbestandteile stellen die beiden grossen Buzzer dar, da sie vom Testleiter sehr leicht zu erreichen sind. In diesem Sinne soll es also möglich sein, die beiden oben angesprochenen Punkte (Sprachaufzeichnung und Screenshot) mit diesen Buzzern zu realisieren.

Mit dem einen Buzzer kann eine Sprachaufzeichnung des Testleiters über das an dem Buzzer angeschlossene Mikrofon getätigt werden. Eine einfache Tonaufnahme als WAV-Datei ist dabei eine ausreichende Möglichkeit. Zudem bietet das Windows - Betriebssystem standardmässig den Audiorecorder an, der programmiertechnisch leicht angesprochen werden kann. Hierzu bietet Java vorgefertigte Klassen und Funktionen an. Eine Verknüpfung dieser Komponenten kann also auf einfache Weise realisiert werden:
Mit dem Druck auf den Buzzer wird die Aufzeichnung über den Audiorecorder gestartet, man spricht in das Mikrofon, und die Aufzeichnung wird wieder beendet, indem der Buzzer losgelassen wird.

Der andere Buzzer dient dazu, ein Abbild des Bildschirminhalts vom Probandenrechner zu speichern. Dies deutet darauf hin, dass die Software zur Betreibung der Buzzerbox eine verteilte Anwendung auf zwei verschiedenen Rechnern ist. Der Probandenrechner im Testraum bildet dabei die serverseitige Anwendung, da von hier aus der Screenshot erzeugt und an den Schnittrechner „geliefert“ wird. Schnittrechner und Testrechner sind beide an das Netzwerk angebunden, so dass die nötige Kommunikation über eine einfache TCP/IP – Verbindung realisiert werden kann. Die Anwendung auf dem Schnittrechner (entspricht dem Client) sorgt dann dafür, dass die übertragenen Bytes wieder als Bild zusammengesetzt werden und in einem geeigneten Bildformat gesichert werden. Der Vorgang wird mit dem Drücken auf den Buzzer gestartet, worauf der Client ein Signal in Form einer Anfrage an den Server sendet, die mit dem angeforderten Screenshot beantwortet wird. [Zlotopolski 2006]

Der einzufügende Aufgabenzähler kann über einfache Counter realisiert und die für die einzelnen Aufgaben benötigte Zeit mit einem gewöhnlichen Timer bestimmt werden. Dabei sollten zum einen die Aufgabenanzahl insgesamt und zum anderen die Anzahl der erfolgreich bearbeiteten Aufgaben gezählt werden, um eine statistische Auswertung über mehrere Tests hinweg zu ermöglichen. Hierfür können zwei der kleinen Druckknöpfe auf der Buzzerbox verwendet werden, die den jeweiligen Counter inkrementieren. Eine Zählung der falsch bearbeiteten Aufgaben ist nicht von besonderem Interesse, da die Leistung des Probanden selbst nicht im Vordergrund stehen darf. Laut Definition soll schliesslich die Anwendung getestet werden, nicht aber das eventuelle Unvermögen des Probanden.
Die Ergebnisse der Zählung können bei Abschluss des Tests in eine einfach gehaltene Textdatei geschrieben werden.

Für die Ermittlung der korrekten Aufgaben sollte der weisse Knopf verwendet werden, da die Farbe weiss im Allgemeinen für positives steht.
Benötigt wird noch eine Funktionalität, die die Zähler für den nächsten Test wieder zurücksetzt. Für diesen Zweck bietet sich der rote Knopf an. Dieser Knopf sollte bei jedem Testende betätigt werden, damit die Zähler wieder bei 0 starten, und somit könnte der Knopf gleichzeitig eine weitere Funktion übernehmen. Es wäre für den Auswertenden sicherlich von Vorteil, wenn er alle zu einem Test gehörenden Daten (Sprachaufzeichnungen, Aufgabenauswertung und diverse Screenshots) in einem separaten Ordner erhält, um die Übersichtlichkeit nicht zu verlieren. Eine unstrukturierte Speicherung aller Daten – speziell wenn mehrere Tests hintereinander durchgeführt werden – in einem einzigen Verzeichnis ist für alle Beteiligten unkomfortabel. Daher wäre es besser, bei jedem nachfolgenden Testdurchlauf einen neuen Ordner zu erstellen, in dem die erzeugten Dateien abgelegt werden.

Noch bislang ohne konkrete Funktionalität belegt ist der grüne Knopf. Aber auch für diesen Knopf sind im Laufe der Zeit einige Ideen entstanden. So könnte der grüne Knopf beispielsweise dafür verwendet werden, Störgeräusche zu erzeugen und auf dem Testrechner abspielen zu lassen. Dieses hat den Hintergrund, dass in der Testphase möglichst realistische Bedingungen für den Probanden gelten sollen.
Eine völlig ruhige Arbeitsumgebung entspräche nicht der Normalität. So wird man in der Berufswelt auch durch Ereignisse abgelenkt, die nicht zum unmittelbaren Arbeitsablauf gehören. Hierzu zählt z.B. das Klingeln eines Telefons.

3.6 Ideen und Konzepte zur Ersetzung des VGA – Films

Eine schnellere und flexiblere Auswertung des VGA-Bildschirms lässt sich dadurch erreichen, die bisherige Filmaufnahme des VGAs durch eine Folge von Screenshots (ähnlich wie beim Buzzertool) zu ersetzen. Gerade dieses Bild liefert die meisten Informationen für den Testleiter, da die hauptsächlichen Aktionen hier statt finden.
Das gestochen scharfe Abbild des Testrechnermonitors kann aus technischen Gründen auf die herkömmliche Weise nicht reproduziert werden, da die zugehörigen Video – Signale in analoge Signale umgewandelt werden.

Dieses könnte prinzipiell so ablaufen, dass in bestimmten, geeigneten Zeitabständen ein automatischer Screenshot von dem VGA-Bildschirm erzeugt und weggespeichert wird. Die optimale Zeitspanne beträgt dabei eine halbe Sekunde, da nicht davon ausgegangen wird, dass innerhalb dieser Zeit mehrfach wichtige Aktionen und Ereignisse auf dem Monitor passieren, so dass nichts an Informationen verloren gehen würde.
Hier wäre dasselbe Prinzip wie beim Buzzertool anzuwenden. Allerdings werden die Screenshots nicht per Knopfdruck geschossen, vielmehr sollte ein eingebauter Timer dafür sorgen, dass die Screenshots automatisch erzeugt werden.

Wichtig ist die Art der Speicherung der Bilder. Das Bitmap – Format beispielsweise verbraucht eine Menge Speicher, da hier die Daten nicht komprimiert werden.
Dies würde die Kapazität auf Dauer sprengen, wenn man bedenkt, dass bei einem Test von 10 Minuten Länge etwa 1200 Screenshots angelegt werden.
Geeignetere Varianten wären das PNG – Format oder auch das JPEG – Format, die eine hohe Datenkompression durchführen und den Speicherbedarf somit im Rahmen halten. Zudem liefert das PNG – Format die beste Qualität und kann verlustfrei auch in andere Dokumente (z.B. in PDFs) eingebunden werden.
Alle Bilder sollten mit Angabe der absoluten Zeit gespeichert werden, um das Rendern zu einem Video zu erleichtern.

Anschliessend müssen die einzelnen Screenshots zu einem Filmstreifen zusammengefügt werden, damit wieder ein Film entsteht, der im späteren Verlauf weiter aufbereitet werden kann und im Endclip als Video darstellbar ist. Hierbei müsste natürlich jetzt die Zeitspanne zwischen den einzelnen Bildern eingehalten werden, damit eine vollständige Synchronität zu den anderen Teilfilmen gegeben ist. Jedes Einzelbild muss also exakt so lang angezeigt werden, bis die Zeitspanne zwischen der absoluten Zeit des darauffolgenden Bildes – der absoluten Zeit des aktuellen Bildes verstrichen ist. Erst dann wird das darauffolgende Bild eingeblendet.

Dieses Verfahren macht zum einen das Rendern des VGA-Films überflüssig, da die einzelnen Screenshots als Sequenzen direkt vorliegen. Damit wird eine sofortige Einsicht und Auswertung dieses Filmstreifens möglich. Ausserdem lässt sich möglicherweise Speicherplatz einsparen, was jedoch zu überprüfen ist.

3.7 Konzepte zur automatischen Reporterstellung

Wie in der Anforderungsplanung wäre es wünschenswert, ein Programm zu haben, das nach Ablauf eines Tests aus den bis zu dem Zeitpunkt zur Verfügung stehenden Daten (Screenshots, Daten des Mousecapturers und ähnliches) automatisch einen Vorbericht für die Auswertung generiert. Dadurch lässt sich der Arbeitsaufwand, den der Testleiter nach einer Testserie zu bewältigen hat, deutlich verringern. Er wäre dann nur noch mit der Auswertung der einzelnen Filme beschäftigt.

Nach einem typischen Usabilitytest sind möglicherweise, sofern alle zur Verfügung stehenden Tools eingesetzt werden, ausser den Filmen folgende Daten vorhanden:

- Mousecapturer mit den Mausmetriken
- Screenshots
- Sprachaufzeichnungen des Testleiters
- Aufgabenzählungen
- Daten vom Eyetracker (Heatmap, Gaze Plot, etc…) [Richter 2008]

Ganz besonders wichtig wäre es zunächst einmal, alle Daten eines Tests zu einer Einheit zusammenzuhalten. Deshalb sollten alle Daten in einem Ordner abgelegt werden und mit einem dem Test eindeutig identifizierenden Namen versehen werden. Diesen Ordner könnte man unterhalb eines anderen Ordners, dessen Name die Studie wiederspiegelt, ansiedeln, so dass aus allen zur Untersuchung gehörenden Tests ein Projekt entsteht.
Der Vorteil ist, dass dadurch eine Gegenüberstellung von mehreren gleichartigen Tests möglich wird, und man beispielsweise die Aufgabenzählung eines Tests direkt mit der eines anderen vergleichen könnte.

Diese Daten müssten von dem zu entwickelten Tool erst einmal gesammelt werden bzw. in das Programm hinein geladen werden. Hierbei müssen verschiedene Dateiformate unterstützt werden – XML für Mousecapturer, PNG für die Screenshots, WAV für die Tonaufnahmen und nicht zuletzt TXT für den einfachen Text einer Aufgabenzählung.

Weiterhin sollte das Tool als Ausgabeformat PDF unterstützen, da dieses das gängige Format für Schriftstücke aller Art ist.

Die Generierung eines Reports läuft dann so ab, dass zunächst in einem ersten Schritt ein leeres Dokument erzeugt wird. Hierfür gibt es entsprechende, bereits bereitgestellte APIs von Java oder anderen Programmiersprachen
(Beispiel: die MFC – Klassen und Bibliotheken in C++), die ohne Mühe verwendbar sind und mit denen Dokumente später automatisch nach PDF konvertiert werden können. Dieses Dokument braucht nur einmal erstellt werden, die nächsten Schritte wären dann für jeden einzelnen Test umzusetzen.

Screenshots vom Buzzertool könnten direkt und unbearbeitet als eingebettete Objekte in das Dokument aufgenommen werden.
Die Daten des Mousecapturers liessen sich schöner darstellen als im
XML – Format. Denkbar wäre die Darstellung in Form einer Tabelle, in der spaltenweise die einzelnen Metrikendaten angegeben sind, und die Zeilen in Abhängigkeit zu der im Mousecapturer angegebenen Zeit stehen.
Das heisst, für jede Sekunde ist ein neuer Eintrag in der Tabelle zu finden, in der die entsprechenden Werte eingetragen sind.
Abbildung 3.7.1 zeigt beispielsweise die Darstellung der Aktivitäten mit der linken Maustaste während eines Tests als Diagramm, in Abhängigkeit von den durch den Testleiter gestellten Aufgaben und der benötigten Zeit. [Wolf 2006]

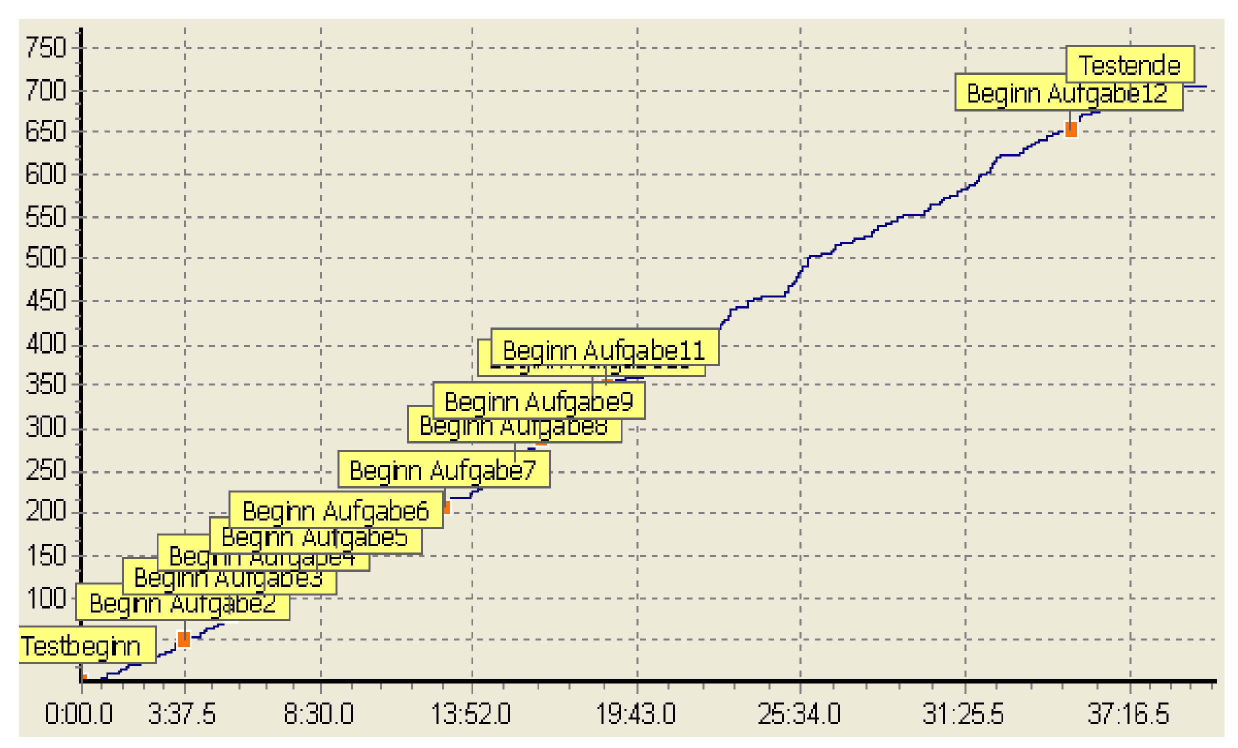


Abb. 3-7-1: Tool von Daniel Wolf – Aktivitäten der linken Maustaste (Diagramm)

Eine andere Möglichkeit wäre es, nur die Endresultate zu nehmen, die letzten Werte aus der XML – Datei. Diese könnte man dann über mehrere Tests hinweg vergleichen und tabellarisch auflisten. Im oben genannten Vorschlag wäre also die Zeit, die den Zeilennamen bestimmt, durch die Testnummer zu ersetzen.
Entsprechendes lässt sich mit der Aufgabenzählung durchführen.
Tabellen können ebenfalls mittels einfacher Operationen in dem Dokument abgelegt werden.

Schwieriger wird der Umgang mit den Sprachaufzeichnungen. Die optimale Idee wäre es, aus dem gesprochenen Wort des Testleiters ein geschriebenes im Dokument zu machen. Worterkennungsprogramme sind zwar bereits auf dem Markt vorhanden, müssten jedoch auf Eignung und Integration in das Labor erst noch getestet werden.

Dennoch sollten dem Testleiter von einem einzigen Dokument aus alle wichtigen Dateien und Materialien zugänglich sein, denn das Ziel ist es, dass er statt einer Menge von verschiedenartigen Dateien nur noch ein Dokument erhält, in dem alles gesammelt ist. Vielleicht liesse sich alternativ daher ein Link, ein Verweis auf die Tonaufnahme in dem PDF – Dokument hinterlegen. Der müsste nur noch vom Testleiter angeklickt werden, worauf seine Sprachaufzeichnung automatisch abgespielt werden würde.

Auf diese Weise würde der Report alle möglichen Auswertungsdaten aller Tests einer Untersuchung in einem Dokument vereinen.

Ein erster Ansatz zur Realisierung wurde bereits 2006 mit dem Tool von Daniel Wolf erreicht, mit dem es möglich ist, die gewonnenen Metriken des Mauscapturers weiterzuverarbeiten und auszuwerten. Ausserdem können hier verschiedene Daten in Projekten zusammengehalten werden, jedoch lassen sich mit dem Tool noch nicht alle gesammelten Daten zu einem Dokument aufbereiten. [Wolf 2006]

3.8 Konzept für eine Operating – Oberfläche

Hier sollen die in den Kapiteln 3.2 bis 3.4 vorgestellten Konzepte zusammengefasst und durch einen Mausklick auf einen einzigen Button gestartet werden können.
Dazu ist ein neues Tool zu realisieren, welches prinzipiell drei Funktionalitäten bieten soll:
1.Filmaufnahmen starten
2.Filmaufnahmen stoppen
3.Rendern und Aufräumarbeiten starten (wie in den Anforderungen beschrieben).

Desweiteren sollte das ganze System nach Durchführung dieses Arbeitsschrittes sauber heruntergefahren werden.

Aufgrund der leichten Synchronisationsprobleme mit den Kameras können die Funktionalitäten bezüglich des Aufnahmeprozesses so noch nicht realisiert werden.
Ein Ersatz hierfür ist mit der von der Firma como entwickelten Software zur Fernsteuerung der Capture Suite zwar vorhanden. Allerdings kann nicht in den Quellcode direkt eingegriffen werden, um Erweiterungen oder Verbesserungen vorzunehmen, da dieser Programmcode leider nicht vorliegt.

Die Idee ist deshalb, zunächst nur die Nachbearbeitung von Filmen zusammenzufassen, zu vereinfachen und über eine einzige Benutzerschnittstelle zu steuern. Dies betrifft insbesondere das Rendern und die Archivierung der gedrehten Filme sowie das Hineinstellen einer Kopie in das öffentlich zugängliche Verzeichnis des Usability – Labors, sofern es irgendwann bereitgestellt ist.
Die einzelnen Konzepte zu den jeweiligen Schritten können in den vorangegangenen Kapiteln nachgelesen werden.

Zu entwickeln ist daher eine Oberfläche, über die die einzelnen Schritte gesteuert werden könnte. Diese Oberfläche verknüpft also die drei Komponenten Skripterstellung/Rendern, Archivierung und Dateitransfer (Abbildung 3-8-1).

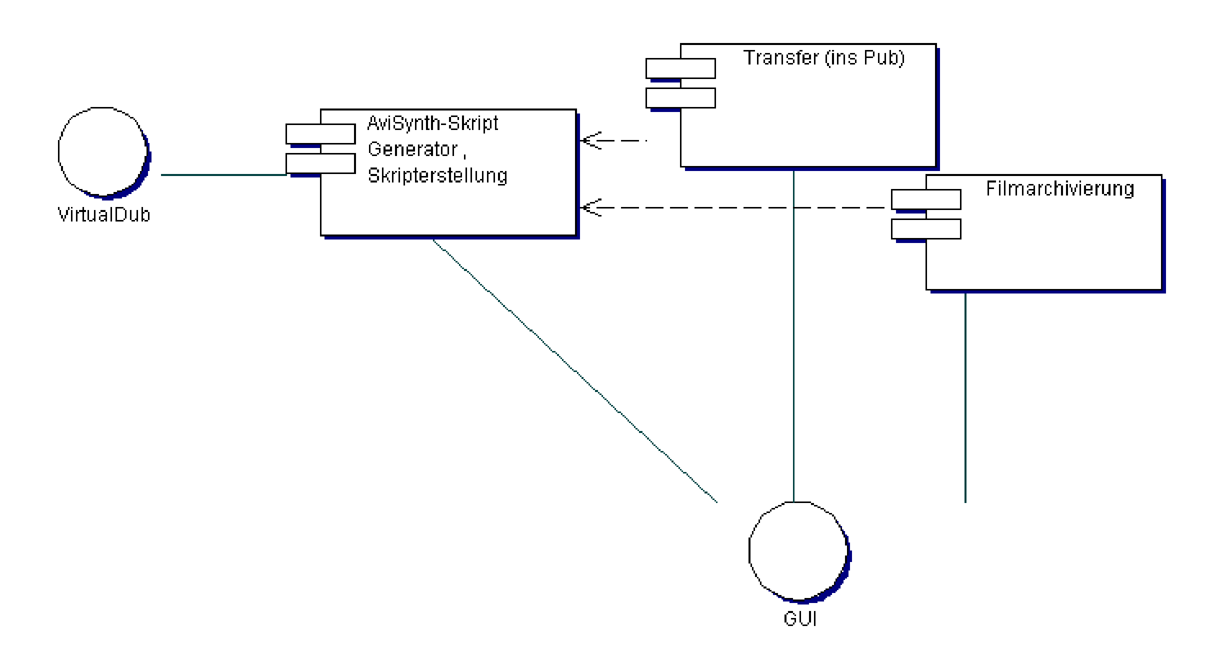


Abb. 3-8-1: Komponenten des Operatingtools

Die Komponente „AviSynth-Skript-Generator“ wiederum muss einen Mechanismus haben, um die fertigen Skripte mit VirtualDub rendern zu können. Hier bietet sich eine einfache Ausführung eines Kommandozeilenbefehls an, mit dem VirtualDub gestartet wird, und über den erforderliche Parameter wie Eingabeskript und Ausgabedatei (entspricht dem fertig gerenderten Film) an VirtualDub übergeben werden.

Ausserdem muss beachtet werden, dass die Komponenten Dateitransfer und Filmarchivierung mit dem Rendern in einer bestimmten Abhängigkeit stehen.
Diese Workflows können nämlich erst dann gestartet werden, wenn es überhaupt fertig gerenderte Filme gibt, die archiviert und gesichert werden müssten.

Letztendlich sollte das Tool folgenden Ablauf garantieren:
1. Erstellung aller benötigten AviSynth – Skripte
2. Rendern der Filme mit den in 1.erstellten Skripten
3. Sichern und Archivieren der Filme auf der Linkstation
4. Weiterleiten der Filme ins öffentliche Verzeichnis
5. Herunterfahren des Rechners

Das automatische Herunterfahren des Rechners mit einzubauen ist insofern sinnvoll, da üblicherweise die Filmaufbereitung am Ende eines Tages stattfindet und – je nach Anzahl der Filme – unter Umständen über Nacht läuft. Ein unnötiges Laufen der Rechner wäre Stromverschwendung und ist daher zu vermeiden. Zudem sollte nach einer Praktikumsveranstaltung oder einer Projektuntersuchung, nachdem alle
Render – Aufgaben klar und formuliert sind, kein weiterer administrativer Aufwand entstehen, also kein Erscheinen des Administrators, nur um Filme auszuhändigen und Rechner herunterzufahren.

Voraussetzung ist allerdings unter anderem die Skriptfähigkeit sämtlicher beteiligter Programme.

Die bereits vorhandenen Implementierungen der Einzelkomponenten können übernommen werden. Es muss nur lediglich möglich sein, diese über die Benutzerschnittstelle ausführen zu können.

3.9 Konzept zur flexibleren Anzeige von fertigen Filmen

Wie angesprochen soll eine Möglichkeit für eine flexiblere Anzeige der gedrehten Videoclips geschaffen werden, die es erlaubt, statt wie bisher einen einzelnen Film mit 6 Kamerasichten und dem VGA – Bild, alle Kamerasichten einzeln (jedoch gleichzeitig) im kleinen Format laufen zu lassen, und auf Wunsch eine bestimmte Kamerasicht in einem grossen Ausschnitt zu zeigen.

In diesem Sinne müsste ein neuer Videoplayer entwickelt werden, mit dem sich mehrere Einzelfilme parallel abspielen lassen. Ein erster Ansatz dazu ist bereits mit dem Auswertungstool von Daniel Wolf vorhanden [Wolf 2006]. Jedoch werden von diesem Tool keine MPEG2 – Dateien unterstützt. Der erste Schritt wäre somit eine Anpassung des Tools, so dass dieses Defizit behoben ist. Die Schwierigkeit hier besteht darin, dass zwar der Quellcode dieses Tools vorliegt, aber das Programm in einer Programmiersprache geschrieben wurde (Delphi), die nicht in den Räumen der HAW verfügbar ist. Zudem existiert leider keine lizenzfreie Version dieser Sprache.
Desweiteren ist Delphi keine Sprache, die innerhalb einer Lehrveranstaltung unterrichtet wird. Insofern ist eine selbstständige Einarbeitung in diese Sprache unumgänglich, oder aber die Weiterentwicklung des Tools sollte denjenigen vorbehalten sein, die bereits Erfahrung mit Delphi haben.
Abhilfe könnte auch eine Portierung beispielsweise nach einer javaspezifischen Plattformumgebung wie Eclipse schaffen.

Die Darstellung innerhalb des Videoplayers könnte dann so abgeändert werden, dass die Ansicht in zwei Bereiche aufgeteilt ist. Die linke Seite (etwa ¾ der Breite) könnte zum Anzeigen der ausgewählten Perspektive genutzt werden, mit den Funktionen, wie sie jeder Videoplayer (wie VLC oder der Windows Media Player) anbietet und die auch schon im Videotool vorhanden sind.

Schwieriger wird der andere Bereich. Hier könnten sämtliche Perspektiven parallel als Einzelfilme ablaufen. Die Einzelfilme werden im kleineren Format visuell in einer Liste dargestellt, die mit einer Scrollbar verschoben werden kann
(siehe Abbildung 3– 9 –1). Klickt man nun auf einen dieser in der Liste dargestellten Perspektiven, so wird diese Ansicht vergrössert in Hauptbereich angezeigt. Die Filmansichten müssten dazu threadgesteuert implementiert werden, da sie alle auf Steuerelemente wie Starten oder Stoppen des Clips gleichzeitig reagieren müssen.

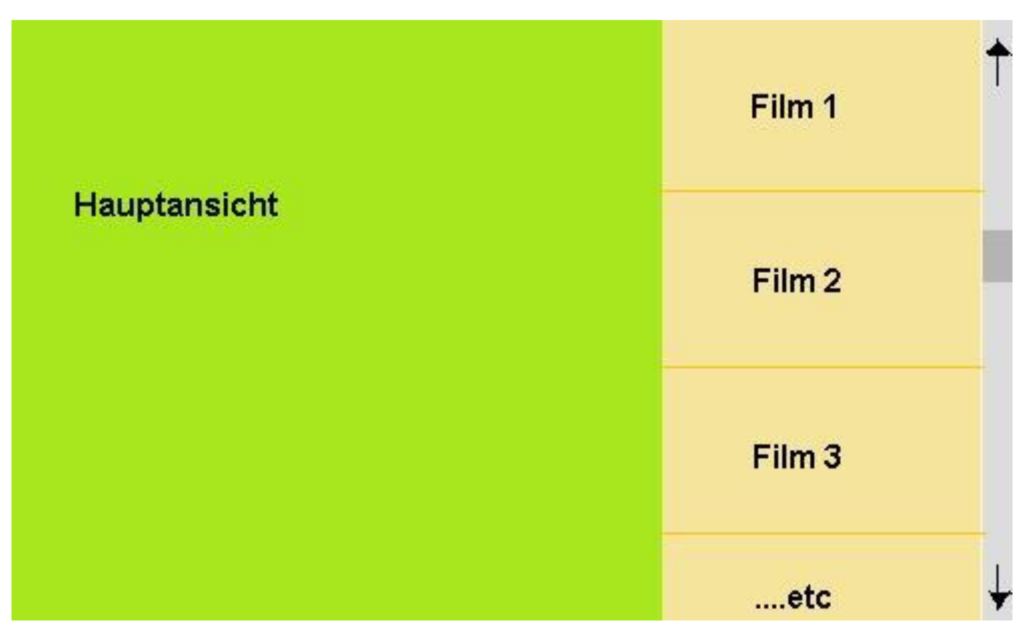


Abb. 3-9-1: Flexiblere Anzeige des Videoclips (Skizze)

Dadurch würden dem Betrachter die Entscheidung selbst überlassen werden, zu welchem Zeitpunkt er welchen Einzelfilm in der Grossansicht ansehen möchte. Dies würde somit auch eine genauere Auswertung ermöglichen, da eine vergrösserte Ansicht von der Qualität her ein besseres Bild darstellt und mehr auf ihm zu erkennen ist als bisher in den kleinformatigen Filmen.

Wie in der Arbeit [Wolf 2006] angesprochen, sollten die Benutzerinteraktionen beim Bedienen des Werkzeugs auch als Marken in einer Schnittliste interpretiert werden. Danach ist ein skriptgesteuerter Filmschnitt auf Basis derartiger Schnittlisten zu realisieren.

Kapitel 4 – Realisierung

Basierend auf den in Kapitel 3 vorgeschlagenen Konzepten kann nun den Anforderungen mit funktionsfähigen Programmen entsprochen werden.
Dieses Kapitel beschreibt die tatsächlich vorgenommenen Realisierungen, soweit sie innerhalb des verfügbaren Zeitrahmens möglich waren.

4.1 Mousetrace

Der Mousetracer stellt ein kleines Tool dar, mit dem es möglich ist, ähnlich wie beim Eyetracking die einzelnen Mausbewegungen zu visualisieren und auszuwerten.
Als Eingabe dient eine XML – Datei, die von dem bereits vorhandenen Mousetracker im Verlaufe eines Tests generiert wird, und in der je halbe Sekunde Einträge über verschiedene Mausdaten enthalten sind.
Aus dieser Datei werden dann die für die Darstellung der Mausverläufe relevanten Daten extrahiert und zu Bildern umgewandelt.

Wie in Abbildung 4 – 1 – 1 zu sehen ist, besteht der Mousetracer aus drei Komponenten. Die Komponente Mousedata übernimmt die Verarbeitung der Eingabedatei und verwaltet die Masse an Mausdaten, die in einem Test gesammelt wurden. Diese Daten werden dann von der zweiten Komponente Graphic benutzt, um daraus einzelne Bilder der Mauspositionen zu erstellen.

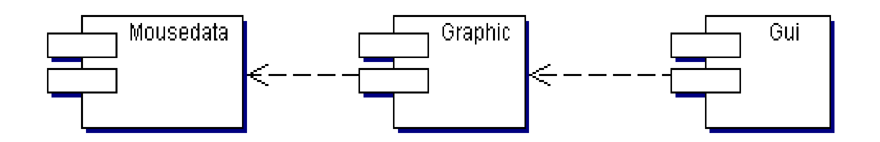


Abb. 4-1-1: Komponenten des Mousetracers

Für die Darstellung wurden zwei Sichtweisen ausgewählt und implementiert. Zum einen wird für jedes einzelne Datenobjekt ein einfaches Bild gezeichnet, das den Mausverlauf wiedergibt und sowohl die zu dem jeweiligen Zeitpunkt aktuelle Mausposition als auch die vorherige Mausposition in Form von Punkten anzeigt. Legt man am Ende alle Bilder aneinander, so lässt sich der gesamte Weg rekonstruieren, den die Testperson mit der Maus zurückgelegt hat. Dadurch lassen sich Rückschlüsse auf das Verhalten des Benutzers im Umgang mit der Maus ziehen.
So könnte unter anderem belegt werden, ob der Proband mit der Maus vielleicht nach einem bestimmten Punkt auf dem Bildschirm gesucht hat und wie lange es dauert, bis die Testperson das gesuchte Objekt gefunden hat.

Verweilt die Maus an einer Position länger, so wird der zugehörige Punkt vergrössert dargestellt. Dies kann beispielsweise ein Hinweis darauf sein, dass das mit der Maus auf dem Bildschirm anvisierte Objekt für den Probanden von besonderem Interesse war.

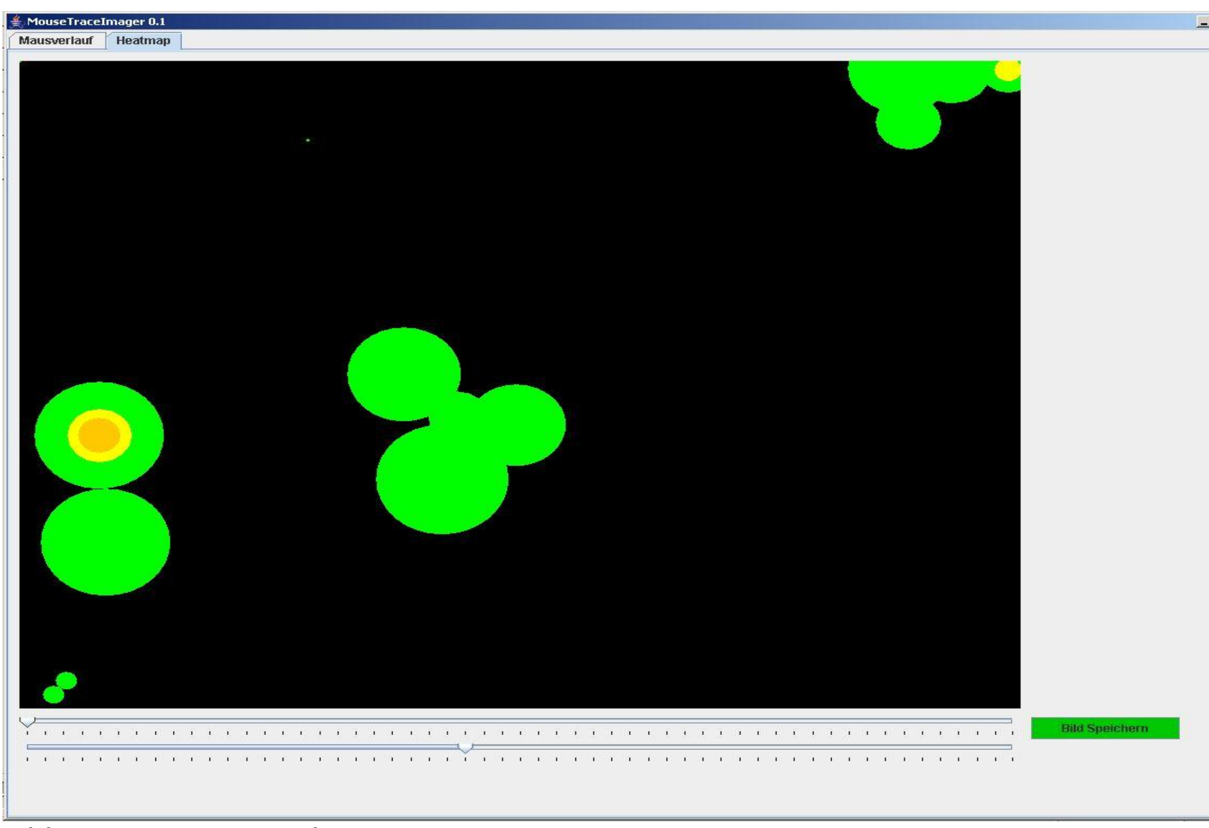


Abb. 4-1-2: Heatmap im Mousetracer

Die zweite Sichtweise ist die Darstellung einer sogenannten Heatmap (siehe Abbildung 4 – 1 - 2). Hier werden alle Mausdaten, die in ein bestimmtes, vordefiniertes Zeitintervall fallen, nach Häufungspunkten zusammengefasst, so dass verschiedene Ballungsgebiete entstehen. Zusätzlich werden Positionen je nach Anzahl des Auftretens andersfarbig dargestellt, hier in der Reihenfolge von grün (selten vorgekommen) über gelb und orange nach rot (Benutzer ist mit der Maus sehr häufig in dem eingestellten Zeitrahmen auf dieser Position gewesen). Bereiche, die mit der Maus gar nicht besucht wurden, werden nicht farblich gekennzeichnet. Auch eine solche Darstellung als Heatmap kann dem Testleiter einen Eindruck davon geben, welche Bereiche des Bildschirms den Probanden besonders interessiert haben, und welche Bereiche möglicherweise nicht wahrgenommen wurden.

Die letzte Komponente Gui stellt die Schnittstelle zum Benutzer (in diesem Fall ist dies der Testleiter) dar, nimmt Eingaben des Testleiters entgegen und verarbeitet diese weiter. Auf der Benutzeroberfläche werden schliesslich die verschiedenen Visualisierungen der Mausbewegungen angezeigt, die dann von hier aus gespeichert werden können.

Es folgt komponentenweise eine kurze Beschreibung der einzelnen Klassen, die für dieses Tool entwickelt wurden. Die einzelnen Pfeile innerhalb der UML – Diagramme stehen dabei für eine Aggregationsbeziehung zwischen den beteiligten Klassen.

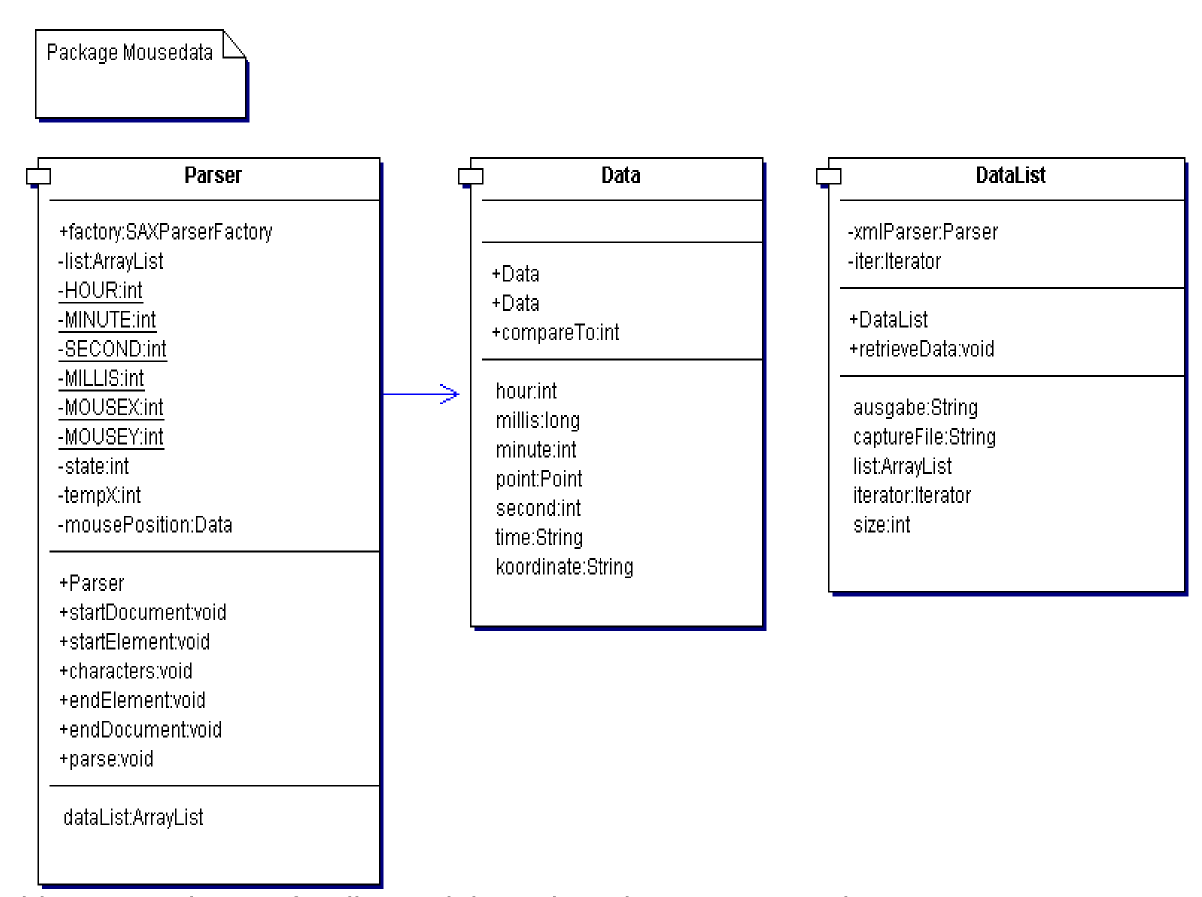


Abb. 4-1-3: Klassen für die Ermittlung der relevanten Mausdaten

Komponente „Mousedata“

Abbildung 4 – 1 – 3 zeigt die einzelnen Klassen, die für die Realisierung der Mausdatenkomponente verwendet wurden. Es gibt hier drei Klassen, die für die Ermittlung der relevanten Mausdaten verantwortlich sind.

Wie bereits angesprochen, werden die Mausdaten aus einer XML – Datei eingelesen. Diese XML – Datei enthält in zeitlichen Abständen von jeweils einer halben Sekunde Einträge von Mausmetriken. Diese Einträge bestehen aus der Zeit seit Beginn des Tests, der Mausposition in auf der X – und Y – Achse des Monitors (wobei der Punkt 0,0 als der erste Punkt links oben auf dem Bildschirm anzusehen ist), die zurückgelegte Strecke gegliedert in Kilometern, Metern bis zu Millimetern, Anzahl Mausklicks mit der rechten, der mittleren sowie der linken Maustaste, die Anzahl der Aktivitäten mit dem Mausrad sowie die Anzahl von Tastaturanschlägen. Je nach Dauer eines Tests läuft hier eine unterschiedlich grosse Anzahl von Daten des Mousecapturers zusammen.

Für den Mousetracer sind jedoch die jeweilige Mausposition und die zugehörige Zeitmarke relevant. Das bedeutet, dass die Informationen in der XML – Datei aussortiert und nur die wirklich interessanten Daten extrahiert werden müssen.
Für das Herausziehen von Daten aus einer XML – Datei gibt es verschiedene Verfahren. Hier wurde ein SAX – Parser eingesetzt, der von der Programmiersprache Java bereits als Klasse zur Verwendung angeboten wird. Da es sich dabei jedoch um eine abstrakte Klasse handelt, musste zu diesem Zweck eine Unterklasse entwickelt werden, die sich hier Parser nennt.
Dabei wird ein Dokument sequentiell von vorn nach hinten nach den einzelnen Elementen (auch Knoten genannt) durchsucht und deren Attribut je nach Anwendungszweck weiterverarbeitet.
Gestartet wird der Vorgang mittels der Methode parse(), der als Parameter die zu durchsuchende Datei mitgegeben wird.

Die X-Position der Maus zu einem bestimmten Zeitpunkt sieht in der XML – Datei beispielsweise so aus:

<mouse-x> 180 </mouse-x>

Mit <mouse-x> wird das Element geöffnet, es folgt der zugehörige Wert, und anschliessend wird das Element mit </mouse-x> wieder geschlossen.

Ein vollständiger Eintrag sieht in der Mousecapturerdatei wie folgt aus:

```
<mouse-data>
        <hour>0</hour>
        <min>0</min>
        <sec>0</sec>
        <mil>500</mil>
        <km>0</km>
        <m>0</m>
        <cm>9</cm>
        <mm>0</mm>
        <pos-x>0</pos-x>
        <pos-y>0</pos-y>
        <left-clicks>0</left-clicks>
        <middle-clicks>0</middle-clicks>
        <right-clicks>0</right-clicks>
        <mouse-wheel>0</mouse-wheel>
        <key-pressed>0</key-pressed>
</mouse-data>
```

Wird beim Parsen nun ein sich öffnendes Element gefunden, wird die Methode startElement() des Parsers aufgerufen. In der Methode characters() werden die Werte des Elements eingelesen, und endElement() kennzeichnet, dass ein sich schliessendes Element (zu erkennen an dem Zeichen „ / “) beim Durchsuchen der Datei gefunden wurde.

Eine Einheit von Metrikendaten wird vom Element „mouse-data“ zusammengehalten. Wird ein solches Element gefunden, so wird ein neues Datenobjekt in der Parserklasse angelegt, das die relevanten Attribute speichert.

Dieses Datenobjekt ist ein Objekt der Klasse Data, eine Klasse, die die relevanten Mausdaten zu einem bestimmten Zeitpunkt erfasst. Ein Datenobjekt besteht dabei

aus der Zeitangabe (Stunde + Minute + Sekunde + Millisekunde) sowie die zu diesem Zeitpunkt vorliegenden X – und Y – Koordinaten der Maus. Um die spätere Verwendung beim Zeichnen zu erleichtern, werden diese Koordinaten als Datentyp Point (entspricht einem Punkt auf dem Bildschirm) gesichert.

Es werden mehrere Data – Objekte nacheinander angelegt und in einer Liste eingetragen. Dadurch wird die sequentielle Abarbeitung der einzelnen Daten bezüglich des Zeitverlaufs sichergestellt.
Die Klasse DataList besitzt so eine Liste. Sie ist letztendlich für das Sammeln der Mausdaten verantwortlich, und aus dieser Liste werden später die Bilder produziert. Sie bildet somit letztendlich die Schnittstelle zu den anderen beiden Komponenten, da sowohl die Graphikkomponente als auch die Benutzeroberfläche Kenntnis über diese Liste haben müssen.
Auch der Parsevorgang wird von dieser Klasse aus über die Methode retrieveData() angestossen.

Komponente „Graphic“

Die Komponente Graphic ist für die Erstellung der einzelnen Bilder verantwortlich. Für beide Visualisierungsarten (Mausverlauf und Heatmap) gibt es dabei jeweils eine eigene Klasse. Zusätzlich wurde eine Klasse MousePoint für die Definition des Mauspunktes implementiert. Sie ist von der Point – Klasse aus Java abgeleitet und enthält zusätzlich noch das Attribut „radius“, um Punkte unterschiedlich gross zeichnen zu können.

Die Klasse MousePicture ist für die Erstellung eines Bildes während der Rekonstruktion des Mausverlaufs zuständig. Entsprechend den Abständen in der XML – Datei wird alle halbe Sekunde ein neues Bild von dem Objekt dieser Klasse gezeichnet. Als Hintergrundbild wird standardmässig ein schwarzes Bild gezeichnet, es kann jedoch auch das Originalvideo des VGA – Films als Hintergrund eingestellt werden. Da die Anzeige auf der Benutzeroberfläche nicht den Massen des Originalbildes entspricht, müssen Höhe und Breite des Originals gesichert werden. Die Massen werden auch die vorgegebene Grösse der GUI skaliert. Pro Bild wird die aktuelle Mausposition als angesprochener MausPoint gezeichnet, zusätzlich wird die Position aus dem letzten Bild mit eingezeichnet. Beide Punkte werden durch eine Linie verbunden. Entspricht der Mauspunkt dem der letzten Position, so wird der Radius des Kreises erhöht. Dementsprechend erscheint der Punkt auf dem Bild grösser. Für das Zeichnen der in blauer Farbe gehaltenen Kreise sorgt die Funktion PaintBild(), das komplette Bild mitsamt Hintergrund wird danach durch den Aufruf der Graphics2D – Funktion paint() bewerkstelligt. Die Methode paint() wurde überschrieben, da es sich hier um ein aus zwei Bildern zusammengesetztes Einzelbild handelt. Ein vollständiges Bild besteht im Mousetracer aus Hintergrundbild sowie den auf transparentem Hintergrund gezeichneten Kreiselementen. Der Vorteil daran ist, dass das durch den Mousetracer gezeichnete Bild später separat gespeichert werden kann und über andere Bilder transparent gelegt werden kann, um besser Vergleiche vornehmen zu können. Ein Beispiel hierfür ist das Aufeinanderlegen eines Mausbildes mit dem zeitlich äquivalenten Eyetrackerbild, wodurch das Zusammenspiel zwischen Auge und Maus untersucht werden kann (vgl. Diplomarbeit Stefan Richter).

Befindet sich die Anwendung im Export – Modus, wird zusätzlich jedes gezeichnete Bild automatisch gespeichert. Dafür wurde die Funktion saveImage() hinzugefügt. Eine Übersicht kann der Abbildung 4 – 1 – 4 entnommen werden, welches die Klassendiagramme der in der Komponenten vorkommenden Klassen wiedergibt.

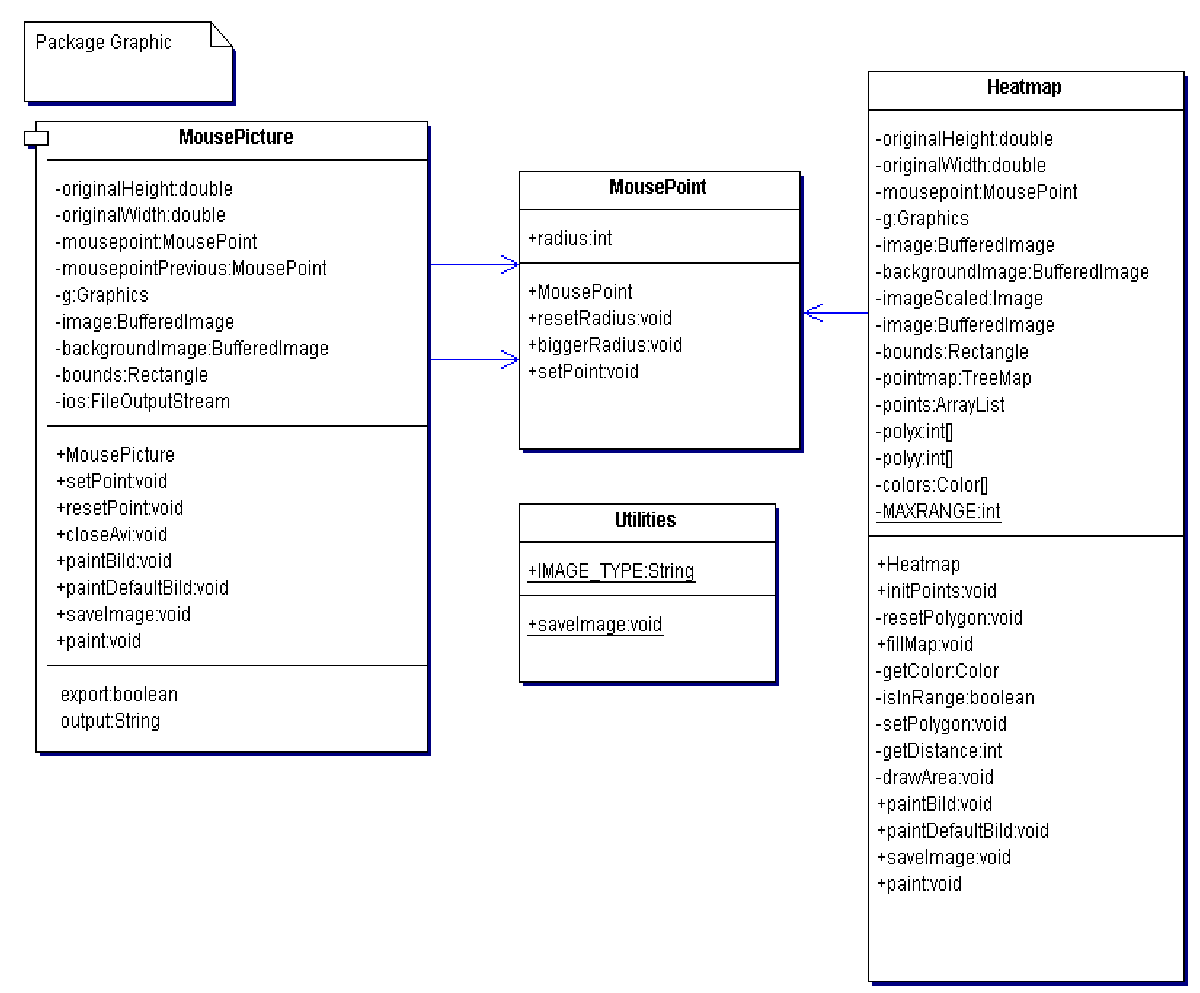


Abb. 4-1-4: Klassen für die graphische Darstellung

Ähnlich funktioniert die Klasse Heatmap für die Darstellung der Ballungsbereiche, in denen sich die Maus aufgehalten hat. Der Zeichenalgorithmus ist hier jedoch ein anderer. Abhängig von dem an der GUI eingestellten Zeitintervall werden hier zunächst alle Mauspositionen dieses Intervalls in einer TreeMap abgespeichert.
Als Schlüssel für die Einträge in dieser Map dienen die einzelnen Positionen, mehrfach auftretende gleiche Positionen können jedoch nicht auftreten. Stattdessen wird der zu dem Schlüssel gehörende Wert jeweils um eine Einheit erhöht. Dadurch kann ermittelt werden, welche Positionen wie oft in der vorgegebenen Zeit besucht wurden.
In Abhängigkeit von der maximalen Entfernung, die zwischen zwei Positionen liegen darf, um diese Mauspositionen ein und demselben Häufungsgebiet zuzuordnen, wird nun Punkt für Punkt durchgegangen, welche Positionen in unmittelbarer Entfernung zueinander liegen. Die maximale Entfernung (MAXRANGE) ist momentan bei 200 Pixel eingestellt.
Wurde ein solches Häufungsgebiet definiert, wird anschliessend aus den zum Bereich gehörenden Punkten ein Polygon gezeichnet. Sind beispielsweise 10 Punkte

in unmittelbarer Entfernung zueinander, so handelt es sich bei dem Polygon um ein Elfeck, da der letzte Punkt zusätzlich wieder mit dem allerersten Punkt verbunden wird. Die Fläche dieses Polygons bildet dann den Häufungsbereich, da davon ausgegangen werden kann, dass auch alle Punkte innerhalb des Bereiches von der Testperson wahrgenommen wurden.
Gezeichnet werden die Punkte wie im Mausverlauf wieder als MausPoints, wobei der Radius des zugehörigen Kreises diesmal der Hälfte der Entfernung zum benachbarten Punkt entspricht. Die Entfernung zwischen zwei Punkten wird mittels Vektorensubtraktion sowie Bestimmung des Betrags des resultierenden Vektors in der Methode getDistance() ermittelt.
Somit wird die Fläche des Polygons vollständig farblich ausgefüllt. Die Farbe des Kreises richtet sich nach der Anzahl, wie oft die Mausposition innerhalb des Intervalls berührt wurde. Kommt eine Mausposition beispielsweise häufig vor, so erscheint der Kreis in roter Farbe, bei nur einmaligem Auftreten dagegen in grüner Farbe.

Komponente „GUI“

Die Komponente GUI ist für die Benutzeroberfläche zuständig. Sie besteht aus zwei Klassen (MainFrame für die Darstellung der Oberfläche sowie SwapChain für das threadgesteuerte Weiterschalten der Einzelbilder in der Mausverlaufsanzeige) und einer Hilfsklasse für die Ermittlung der geeigneten Dateiformate, die beim Exportieren und Speichern der einzelnen Bilder sowie Laden der Basisdaten verwendet werden. So wird sichergestellt, dass beispielsweise die einzulesende Mousecapturer Datei auch tatsächlich im XML – Format vorliegt.
Standardmässig werden die vom Mousetracer erstellten Bilder im komprimierten PNG – Format abgespeichert.

Die einzelnen Klassen, die in dieser Komponente beinhaltet sind, sind als Klassendiagramm in der Abbildung 4 – 1 – 5 skizziert.

Die Klasse SwapChain ist genauer genommen eine innere Klasse von „MainFrame“. Es handelt sich dabei um einen Thread, der bei der Anzeige des Mausverlaufs dafür sorgt, dass alle halbe Sekunde das nächste Bild der Mausposition angezeigt wird. Gestartet wird der Thread, sobald in der Applikation auf den Button „Start“ gedrückt wird. Ein Klick auf „Stop“ beendet den Thread wieder.

Hauptklasse ist jedoch die Klasse MainFrame. Innerhalb dieser Klasse wird die gesamte Benutzeroberfläche aufgebaut, die Eingaben vom Benutzer entgegengenommen und weiterverarbeitet. Entsprechende Aktionen und Funktionsaufrufe werden an die anderen Klassen von hier aus weitergeleitet bzw. vorgenommen. Da die GUI der zentrale Mittelpunkt der Anwendung ist, in dem zum einen die erforderlichen Basisdaten geladen und zum anderen die Bilder angezeigt werden, entsteht hier eine sehr starke Abhängigkeit mit den anderen Klassen, wie beispielsweise der Heatmap – Klasse oder der MousePicture – Klasse.
Die Abbildung 4 – 1 – 6 zeigt noch einmal alle verwendeten Klassen im Überblick. Dabei wird auch deutlich, welche Klassen hier in direkter Beziehung zueinander stehen, und welche Abhängigkeiten dementsprechend untereinander bestehen.

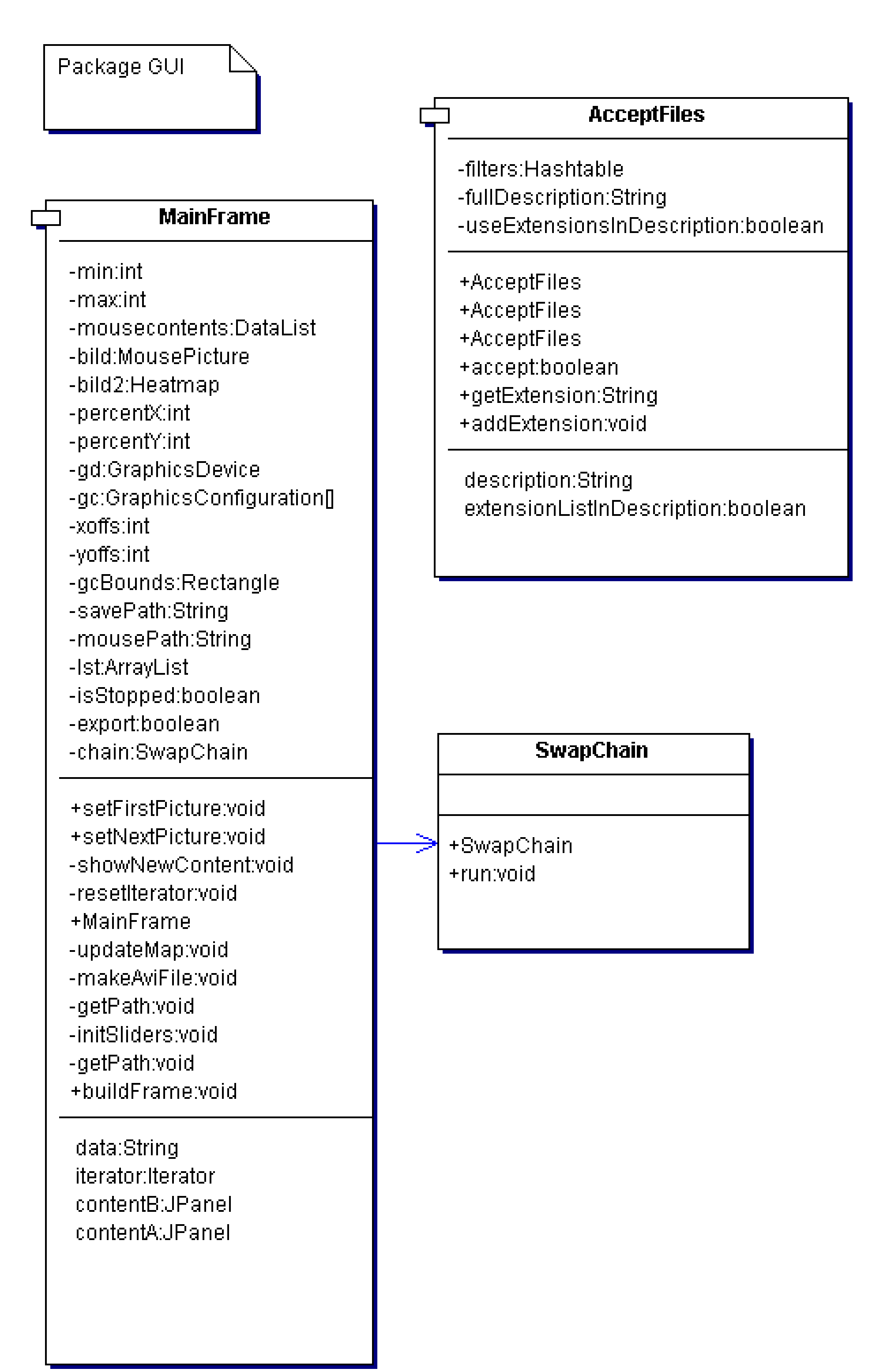


Abb. 4-1-5: Klassen der Benutzeroberfläche

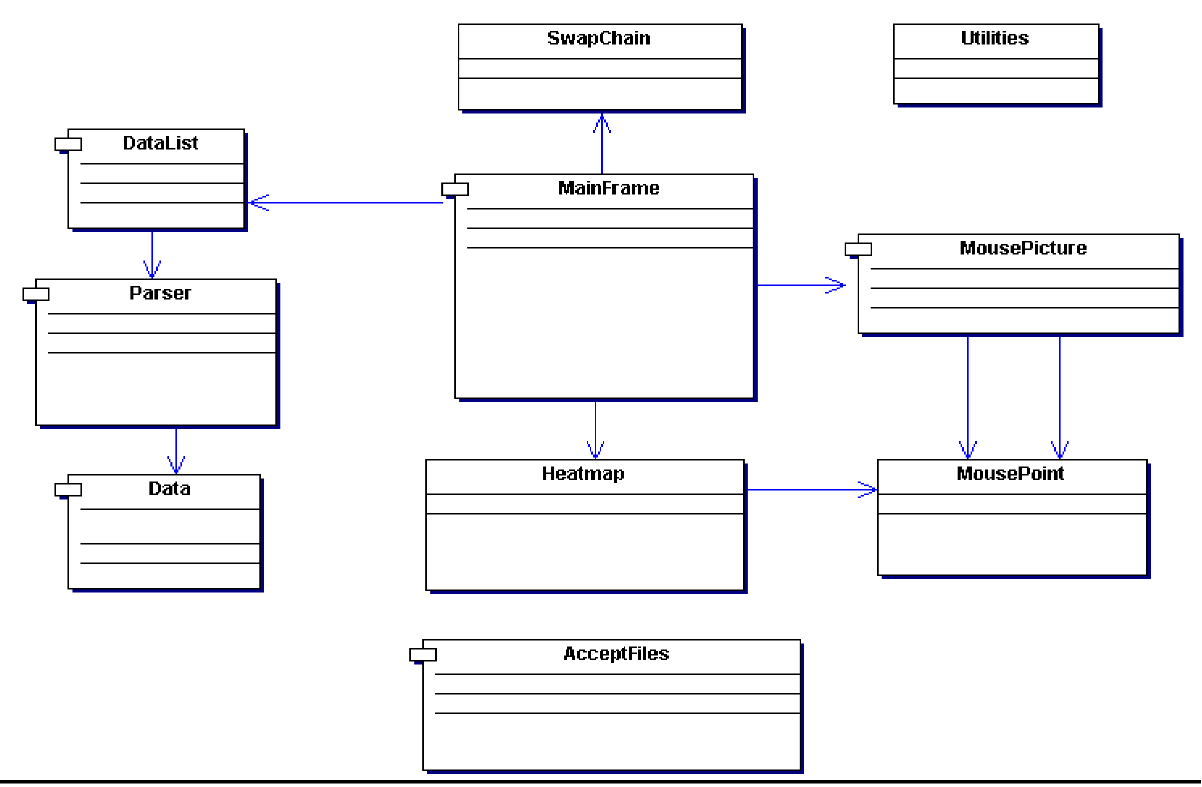


Abb. 4-1-6: Zusammenspiel der einzelnen Klassen

Die GUI selbst ist aufgeteilt in zwei Tabs, jeweils eine für die Anzeige des Mausverlaufs und der Anzeige der Heatmap. Für die Darstellung wird der gesamte Mittelbereich der Oberfläche verwendet, damit die Bilder nicht zu klein werden. Beim Start des Programms wird die verwendete Auflösung der am Rechner angeschlossenen Grafikkarte dynamisch ermittelt und als Massstab für die Massen der Oberfläche verwendet. Somit ist der Mousetracer unabhängig von der eingestellten Bildschirmauflösung und passt sich automatisch jeder Veränderung an.

Nach dem Einlesen der Basisdaten werden alle notwendigen Initialisierungen von der GUI aus vorgenommen. Hierunter fällt das Ermitteln der für den Mousetracer relevanten XML – Daten (über ein zur GUI gehörendes Objekt der Klasse DataList) sowie die Anzeige des allerersten Bildes noch vor dem Start des Verlaufs (geschieht über die Funktion setFirstPicture()).
Die eingelesenen Positionen über den XML – Parser werden dementsprechend so skaliert, dass sie ohne Verlust auf dem dafür vorgesehenen Bereich der Benutzeroberfläche angezeigt werden können.

Der erste Tab „Mausverlauf“ ist für die Anzeige der einzelnen Mauspositionen in Abhängigkeit der verstrichenen Zeit zuständig. Hier befinden sich Buttons zur Steuerung des Verlaufs (Start, Stop, Pause) sowie zum Einlesen der Basisdaten und Speichern des Mausverlaufs als AVI – Datei.
Nach dem Klick auf Start wird wie erwähnt ein Thread gestartet, der alle halbe Sekunde ein Signal zum Zeichnen des nächsten Bildes an die Oberfläche sendet. Das GUI – Objekt wiederum legt dann ein neues MousePicture – Objekt an, von dem aus die betreffenden Aktionen zum Zeichnen ausgeführt werden.

Das bedeutet, dass die einzelnen Bilder dynamisch nacheinander gezeichnet werden während eines Durchlaufs. Dieses wird über die Funktion setNextPicture() angesteuert. Dennoch hat man das Gefühl, man würde ein Video sehen. Einen Eindruck von dieser Funktionalität kann aus der Abbildung 4 – 1 – 7 gewonnen werden.

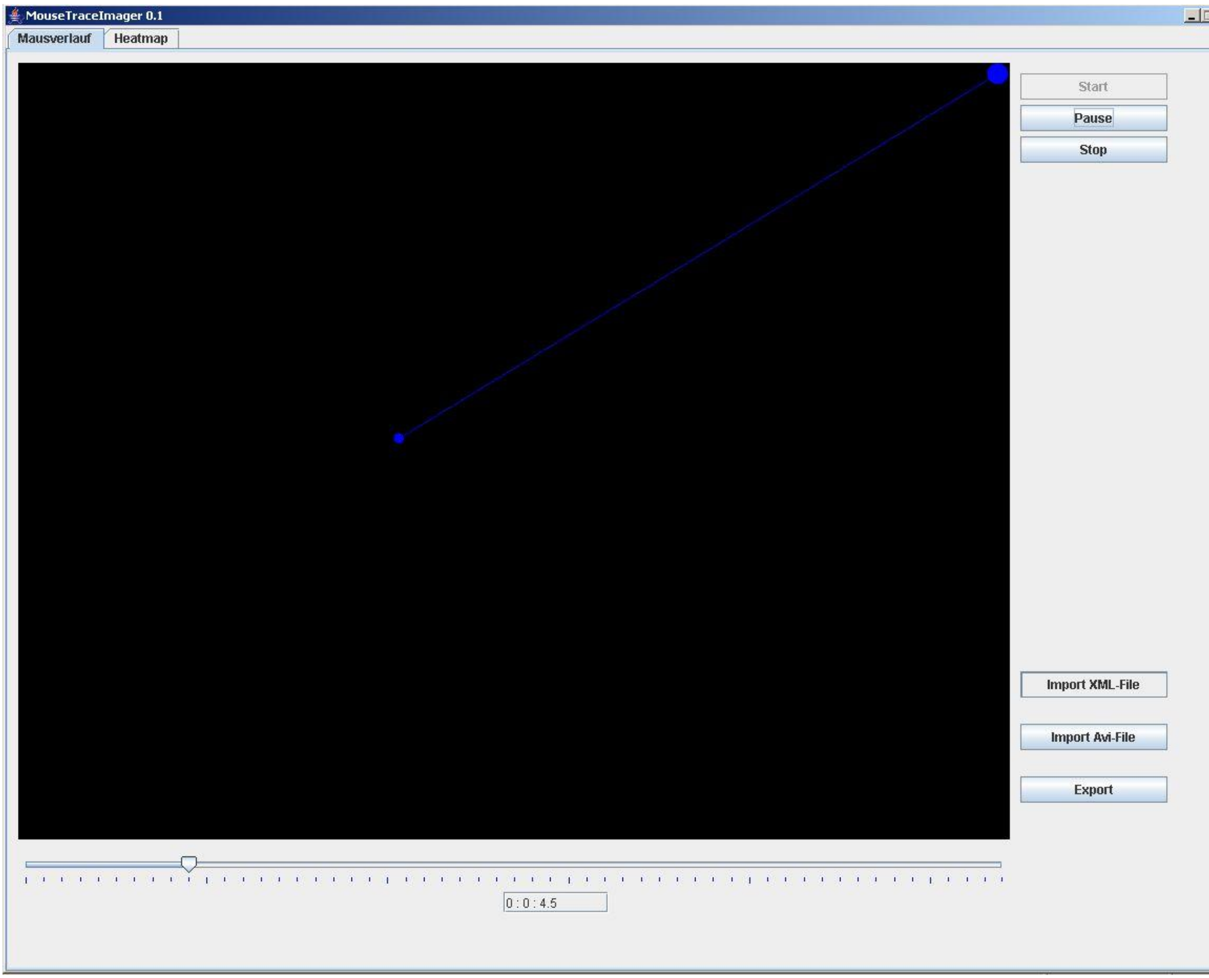


Abb. 4–1–7: Mousetracer in Aktion

Der zweite Tab „Heatmap“ ist für die Anzeige der Heatmap zuständig. Wie eine Heatmap ermittelt und gezeichnet wird, wurde bereits innerhalb der Komponente Graphic erläutert. Die Oberflächenanzeige unterscheidet sich dennoch etwas von der des ersten Tabs, da es möglich sein soll, das zu berücksichtigende Zeitintervall für die Erstellung einer Heatmap frei einstellbar zu machen. Hinzugefügt wurden an dieser Stelle zwei Slider, mit denen die beiden Intervallgrenzen (kleinster Zeitpunkt sowie grösster Zeitpunkt) beliebig verstellt werden können. Die angezeigte Heatmap passt sich den Gegebenheiten automatisch an, d.h. es wird eine neue Heatmap gezeichnet (hierfür steht die Funktion UpdateMap() zur Verfügung). Eine beispielhafte Heatmap ist in der Abbildung B5 – 10 im Anhang B auf der Seite 132 zu sehen.
Über die Speicherfunktion mittels des „Bild speichern“ – Buttons lässt sich jede generierte Heatmap separat als PNG – Datei abspeichern, bei einem Klick wird die entsprechende Funktion des Heatmap – Objekts aufgerufen (saveImage()).

4.2 Rendern

Im folgenden Abschnitt soll zunächst einmal die Implementierung des Programms, welches die AviSynth-Skripte erzeugt, dargestellt werden.
Während der Aufnahme werden die einzelnen Teilfilme von den beiden Capture-Rechnern auf die externe Festplatte (mit dem Laufwerksbuchstaben G:) gespeichert.
Die Aufteilung hierzu sieht folgendermassen aus:

Film Kamera 1 → G:\cam1\
Film Kamera 2 → G:\cam2\
Film Kamera 3 → G:\cam3\
Film Kamera 4 → G:\cam4\
Film Kamera 5 → G:\cam5\
Film Kamera 6 → G:\cam6\
Film VGA-Bildschirm → G:\VGA\
Film Mousecapturer → G:\mouse\

Die Dateinamen der einzelnen Filme werden nach dem Muster „MMTTJJ_h+min+sec_ms_Kameranummer“ vergeben, bestimmt durch den Startzeitpunkt der entsprechenden Aufnahme. Das Muster der Dateinamen ist vom Hersteller vorgegeben und kann nicht verändert werden.
Als Beispiel sei hier eine Datei für die Kamera 1 angegeben:
012308_125722_437_cam1.avi.
(Hier muss angemerkt werden, dass die eigentliche Aufnahme zwar im MPEG-Format stattfindet, da VirtualDub jedoch .AVI-Dateien besser handhaben kann,
wird nach der Aufnahme in einem Zwischenschritt der Rohfilm in das AVI-Format konvertiert. Siehe auch Kapitel 5.3.1 und 5.3.2)
Diese Aufnahme wurde also am 23.01.2008 um 12:57:22 Uhr mit der Kamera 1 gestartet. Werden gleichzeitig mehrere Kameras verwendet, so besitzen die Dateinamen der Rohfilme nahezu identische Bezeichnungen. Allerdings können einzelne Abweichungen im Bereich der Sekundenangabe auftreten, da die Kameras herstellerbedingt nacheinander bis zu einer Sekunde verzögert gestartet werden, und nicht etwa vollständig synchron. Dieses stellt für das Programm jedoch kein Problem dar, da die einzelnen Dateinamen per Substring-Suche miteinander verglichen werden. Sind zwei zu vergelichende Dateinamen bis zu der Stelle, an der die Sekunden angegeben werden, identisch, so kann davon ausgegangen werden, dass beide Rohfilmdateien zu dem gleichen Test gehören. Es ist unwahrscheinlich, dass innerhalb weniger Sekunden ein Test beendet und der nächste gestartet wird.

Der am Ende zur Verfügung stehende, fertig gerenderte Film soll so aufgebaut sein, dass die Kamerasichten alle parallel und nicht zu klein darin abgebildet werden. Momentan ist der Endfilm in zwei Ebenen aufgeteilt (Abbildung 4-2-1):

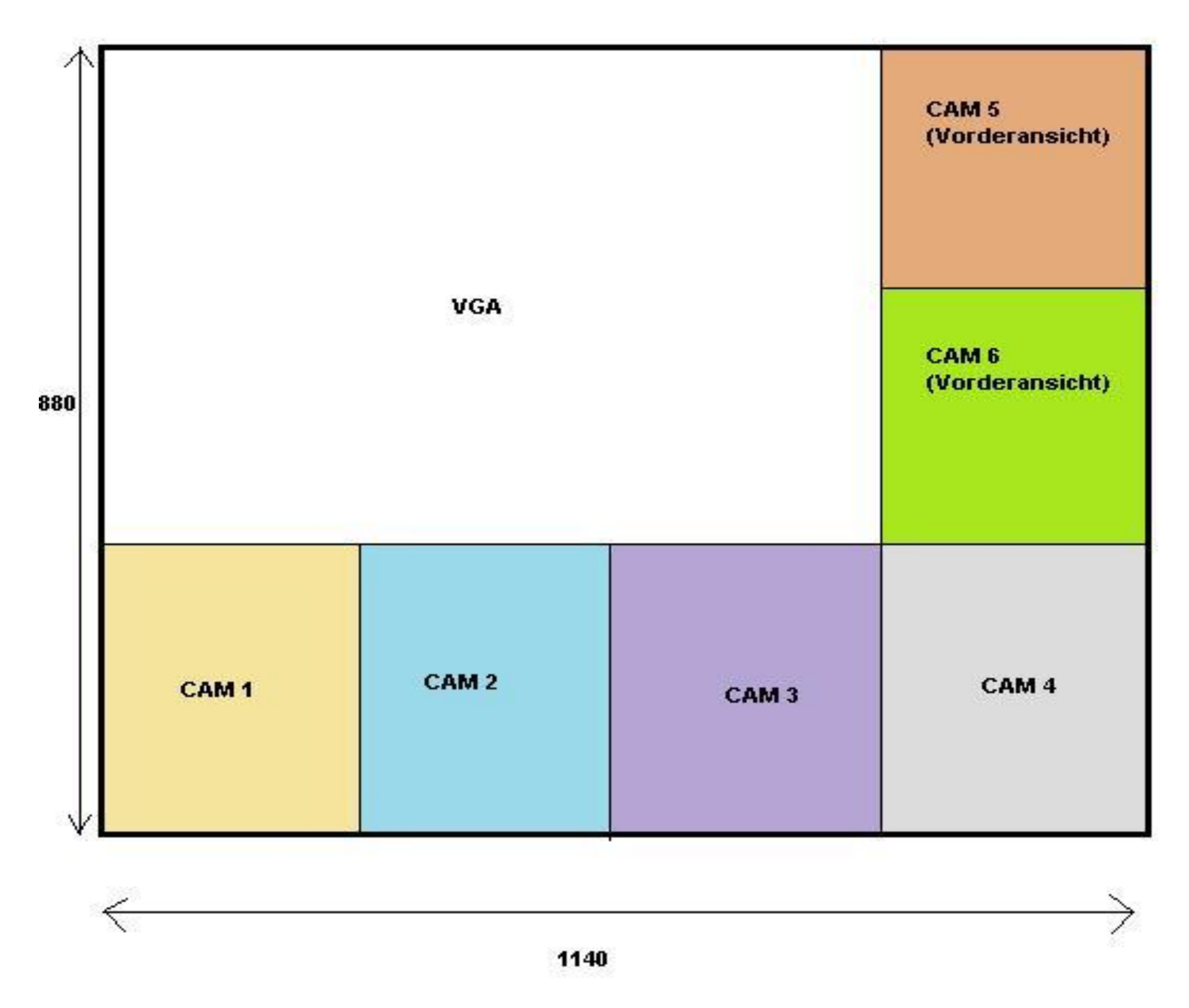


Abb. 4-2-1: Aufteilung der einzelnen Kamerasichten im fertigen Film (4:3)

In der oberen Ebene sind der VGA-Bildschirm sowie die beiden Kameras, die den Probanden von vorn zeigen (Kamera 5 und Kamera 6) platziert, die anderen 4 Kameras bilden die zweite vertikale Ebene. Der meiste Raum wird dabei der Aufzeichnung des VGA-Bildschirms zugesprochen, da hier die hauptsächliche Interaktion zwischen Proband und dem zu testenden System während eines Usability-Tests stattfindet. Das Augenmerk der auswertenden Person wird demnach auch am meisten auf dieser Sicht liegen.

Die oben stehende Abbildung lässt ferner vermuten, dass für die jeweiligen Kamerasichten jeweils ein Raum von fest definierter Grösse zur Verfügung steht.
Lässt man allerdings eine Kamera weg, so wird der für diese Kamera vorgesehene Platz auf die anderen Teilfilme, die sich in der gleichen vertikalen Ebene befinden, aufgeteilt. Beispiel: Für einen Test kommt Kamera 2 nicht zum Einsatz.
Dies bedeutet, dass auf dem Endfilm die Teilfilme der Kameras 1, 3 und 4 etwas breiter erscheinen als in der Abbildung dargestellt, da der nicht benötigte Raum von Kamera 2 gleichmässig auf die anderen Kameras umgeschichtet wird.

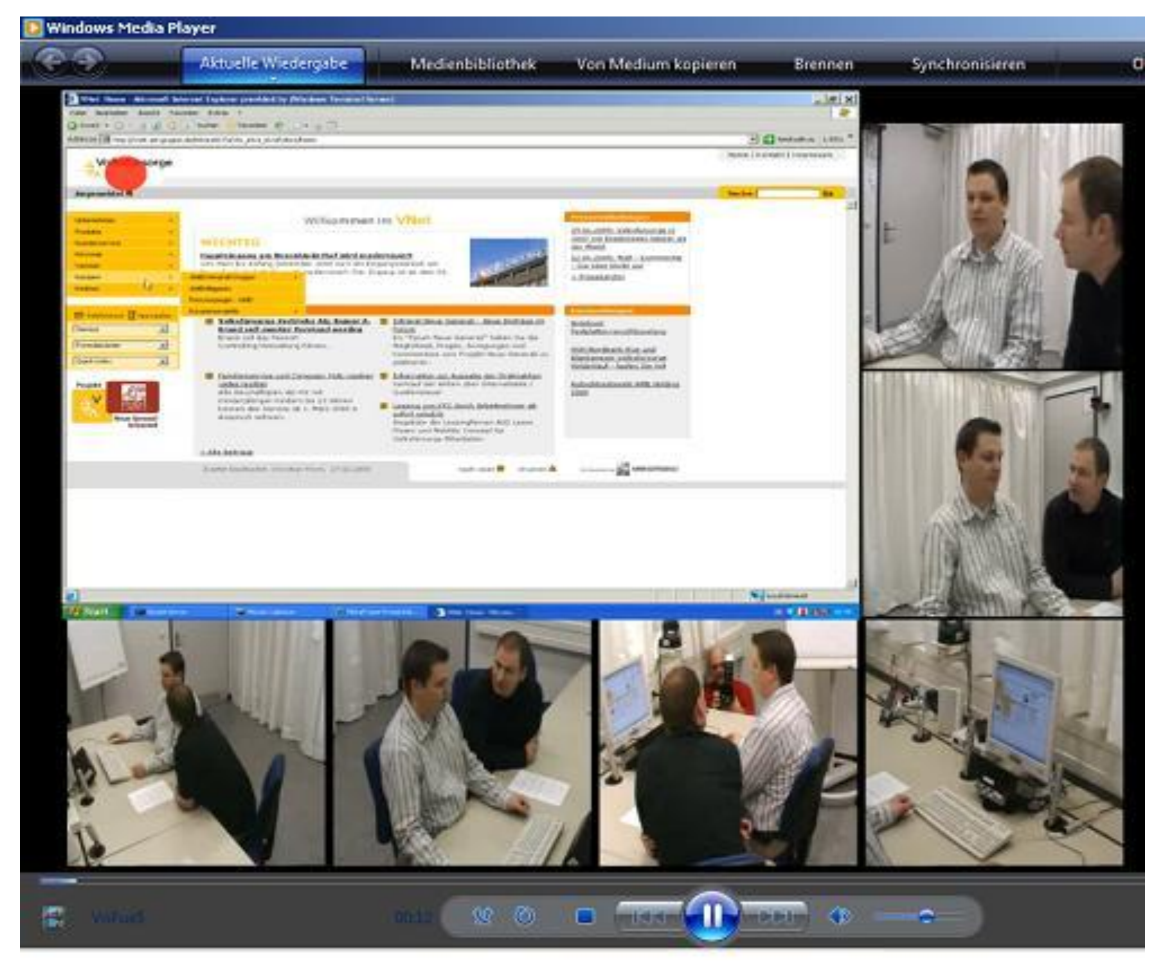


Abb. 4-2-2: Fertiger Film mit allen 6 Kameras und Eyetracker

Letztendlich sind die Anzahl der Teilfilme sowie die genaue Aufteilung dieser Filme im AviSynth-Skript definiert. Wie wird nun ein solches Skript erzeugt ?

Zunächst einmal besteht das zu diesem Zweck entwickelte Programm im Wesentlichen aus zwei Klassen sowie einem Interface. Vorab sei erwähnt, dass das Interface ScriptPattern lediglich einige Zeichenkettenkonstanten des Typs static string enthält. Diese Konstanten beinhalten alle Befehle für Avi-Synth-Skripte,
die bei der Filmerstellung möglich, aber auch für die Zwecke des Labors ausreichend sind. Das heisst, dass sowohl die Skalierung der vertikalen Ebene mit drei als auch die mit vier Kameras dort angegeben sind.
Für die konkreten Dateinamen der einzelnen Teilsysteme werden Platzhalter verwendet (ebenfalls Zeichenketten), die später durch das eigentliche Programm mit den entsprechenden Werten besetzt werden.
Das Programm selbst sucht sich also nur die zu dem Testfilm passenden Befehlsfragmente aus diesem Interface heraus und vervollständigt diese.

Die eigentliche Zusammensetzung des benötigten AviSynth-Skriptes geschieht in der Klasse GenerateScript. Dabei wird so vorgegangen, dass in einem ersten Schritt überprüft wird, welche Kameras denn eigentlich bei dem Usability-Test verwendet wurden. Daraus kann dann gleichzeitig die Gesamtzahl der verwendeten Kameras bestimmt werden. Dieses geschieht mit Hilfe eines simplen Arrays des Typs Boolean, das aus insgesamt acht Werten besteht – für jede Kamerasicht (sowie VGA und Mousecapturer) jeweils ein Wert. Der Wert „true" gibt dabei an, dass die mit dem

Eintrag korrespondierende Kamera bei dem Test zum Einsatz gekommen ist, „false“ steht entsprechend für „nicht verwendet“.
Mit diesen Informationen extrahiert diese Klasse nun aus dem angesprochenen Interface die richtigen Befehlszeilen heraus, um anschliessend für den zu entwickelnden Endfilm ein korrektes Avi-Synth-Skript aufzustellen.
Das Avi-Synth-Skript selbst ist dabei nichts weiter als eine einfache Textdatei.

Die zweite Klasse mit dem Namen All_AvsGen (bedeutet soviel wie „Erzeuger aller AviSynth-Skripte“) nimmt das fertige Avi-Synth-Skript entgegen, vervollständigt dieses mit den konkreten Dateinamen der zu verwendenden Teilfilme und speichert es am Ende in einem eigens dafür angelegten Ordner ab.
Die Bestimmung der richtigen Teilfilme läuft dabei folgendermassen ab:
Es wird davon ausgegangen, dass es immer einen VGA-Film gibt, da dieser der Hauptbestandteil des fertigen Films sein wird. Dies bedeutet, dass der entsprechende Ordner „G:\vga\“, der die VGA-Rohfilme enthält, als Ausgangspunkt für die Ermittlung der anderen Kameras anzusehen ist.
Wird in diesem Ordner nun eine Rohfilmdatei gefunden (diese Datei muss die Endung .avi besitzen), so wird zunächst einmal aus dem Dateinamen eine Zeichenkette für Vergleichszwecke extrahiert. Dieser Substring enthält Datums - und Uhrzeitangaben des Dateinamens. Dann werden alle anderen Filmordner auf der externen Festplatte G:\ nach Dateien durchsucht, die mit dem Vergleichsstring übereinstimmen. Auf diese Weise kann ermittelt werden, welche anderen Teilfilme zu dem VGA-Film gehören. Wird in einem Ordner kein geeigneter Dateiname gefunden, der zu dem Vergleichsstring passt, so wird als Dateiname lediglich ein String „NOTFOUND“ zurückgegeben. Die Dateinamen der gefundenen Teilfilme werden in einem String-Array zwischengespeichert und später für die Angabe im AviSynth-Skript weiterverwendet. „NOTFOUND“ gibt hierbei an, dass die betreffende Kamera bei dieser Aufzeichnung nicht verwendet wurde.

Dieses Vorgehen wird für alle gefunden Dateien im VGA-Ordner wiederholt, d.h, zu jedem existierenden VGA-Film wird ein passendes AviSynth-Skript erzeugt. Somit können mehrere AviSynth-Skripte hintereinander bequem in einem Schritt geschrieben werden.

Abbildung 4-2-3 zeigt ein Sequenzdiagramm des kompletten Programmablaufs.

Zu finden sind die fertigen Skripte in dem Pfad “C:\UselabOrdner\AviSynthSkripte\“.
Es sei darauf hingewiesen, dass dieser Ordner nur für die Skripte verwendet wird und dass ältere Skripte nach Fertigstellung eines Films derzeit noch manuell gelöscht werden müssen, da sonst bei einem späteren Rendervorgang die gleichen Filme nochmal gerendert werden würden. Über ein automatisiertes Löschen nicht mehr benötigter Skripte nach Beendigung des Renderprozesses sollte nachgedacht werden.

Das eigentliche Rendern findet im Anschluss an die Skript-Erzeugung mit dem Programm VirtualDub statt. Näheres hierzu ist im Anhang unter der Bedienungsanleitung von VirtualDub zu finden.

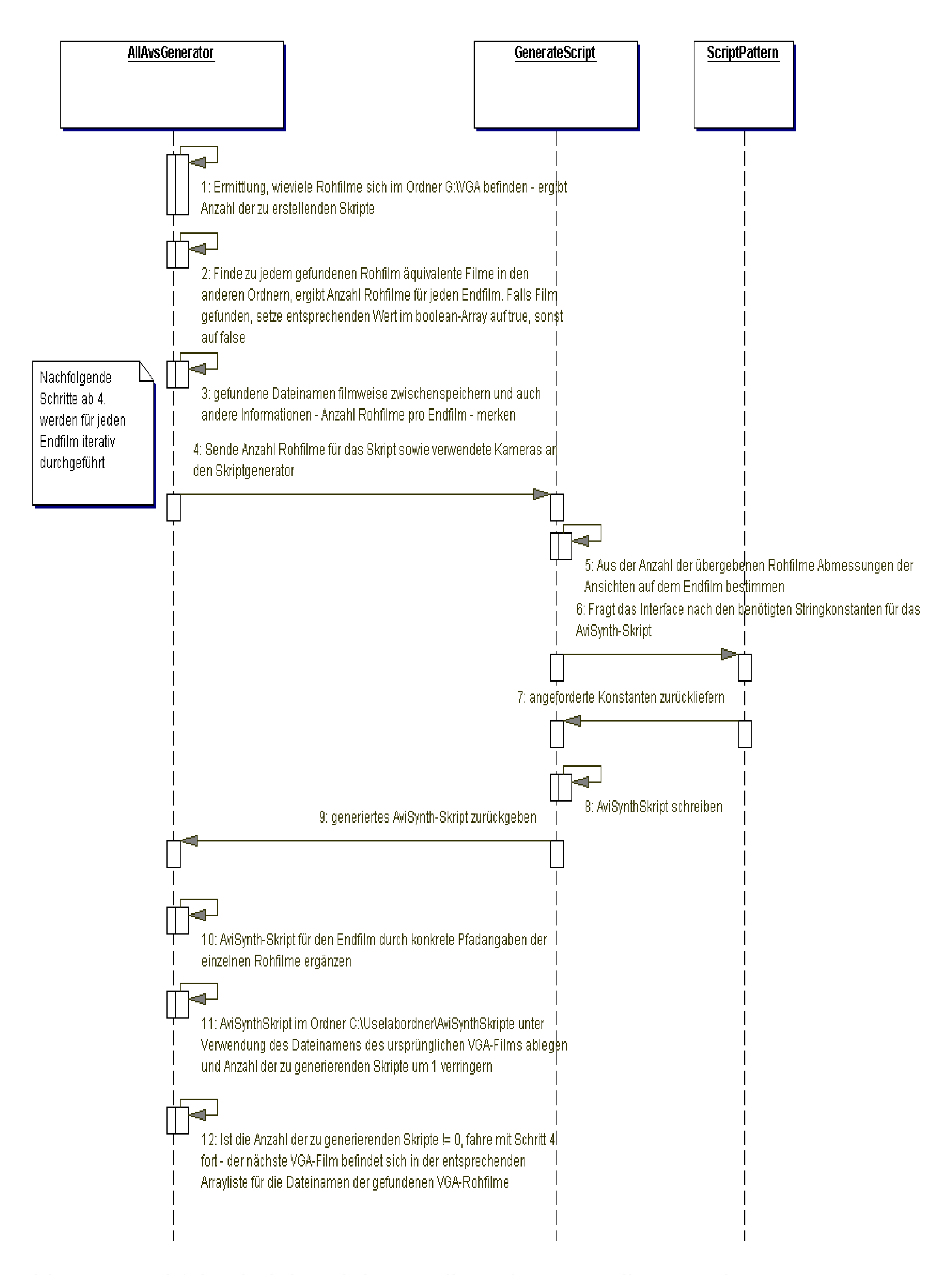


Abb. 4-2-3: Abfolge bei der Skripterstellung (Sequenzdiagramm)

4.3 Filmsicherung und – archivierung

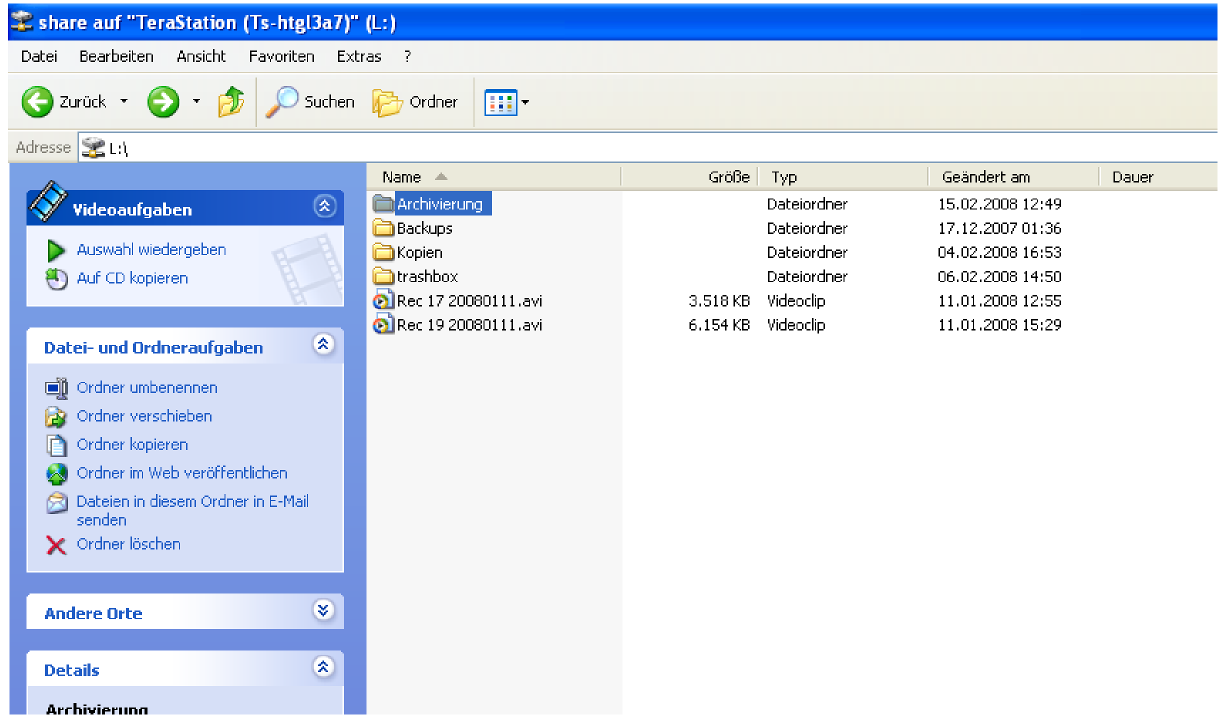


Abb. 4-3-1: Verzeichnisstruktur auf der Linkstation

Wie der obigen Abbildung zu entnehmen ist, wurde auf der Linkstation ein eigenes Verzeichnis „Archivierung“ eingeführt. In diesem Verzeichnis sollen die aufgenommenen Filme gesichert werden.

Aufgrund der Tatsache, dass die Linkstation vom Schnittrechner aus über das Netzwerk mit dem Laufwerksbuchstaben L:\ ansprechbar ist, ist die Implementierung dieser Funktionalität denkbar einfach. Sie sieht wie folgt aus:

Da das Programm kaum Funktionen benötigt, genügt eine einzelne Klasse, die „Filmarchivierung“ genannt wird. Diese Klasse schaut zunächst in den Ordner „Uselabordner\Filme“ nach, ob dort neue, noch nicht gesicherte Filme vorhanden sind. Die Dateinamen dieser Filme werden als Strings in einem Array gespeichert.

Dann wird der Reihe nach der Dateiname auf das Zielverzeichnis angepasst.
Aus „Uselabordner\Filme\Film1.avi“ wird somit „L:\Archivierung\Film1.avi“.
Dies geschieht über einfache Stringoperationen. Gleichzeitig wird der Inhalt der jeweiligen Quelldatei (also Audio – und Videostream) in Form von Bytes in das Zielverzeichnis kopiert. Dies funktioniert jedoch nur, wenn auf das Laufwerk
L:\ zugegriffen werden kann. Ist dies nicht der Fall, so wird eine Fehlermeldung erhoben und das Programm abgebrochen.

Die ursprünglichen Videos werden dabei nicht gelöscht, so dass selbst im Falle eines Programmfehlers die Originaldatei noch erhalten bleibt. Bei Bedarf müssen sie manuell aus dem Filmeordner entfernt werden.

Die zweite Sicherung erfolgt über das Backup – Programm Acronis und ist deshalb hier nicht vorgesehen. Die Bedienungsanleitung dazu kann im Anhang gefunden werden.

4.4 Operating – Oberfläche

Das Operatingtool ist dazu gedacht, die Funktionalitäten AviSynth – Skript – Erstellung, Rendern, Archivieren und Transport in ein von aussen öffentlich zugängliches Verzeichnis zu vereinen, die Umsetzung dieser Funktionalitäten effizienter zu gestalten und zu vereinfachen. Ausserdem werden die einzelnen Schritte hier zusammengefasst, so dass der komplette Vorgang über einen einzigen Mausklick gestartet werden kann.
Die einzelnen Realisierungen sind in den vorangegangenen Kapiteln bereits genauer beschrieben worden und sollen hier nicht noch einmal wiederholt werden.

Hinzu kommt eine Klasse, die die Benutzeroberfläche aufbaut und Aktionen auf dieser entgegennimmt. Die Oberfläche besteht aus sechs Buttons sowie einem Textfeld, in dem Hinweise über den Verlauf und Fortschritt der einzelnen Schritte ausgegeben werden. Die Buttons stehen stellvertretend für die einzelnen Workflows:

- Erstellung der AviSynth – Skripte
- Rendern der Filme
- Filmarchivierung
- Transport ins Pub
- Herunterfahren des Rechners
- Komplettdurchlauf durch alle Funktionalitäten (entspricht dem mehrfach angesprochenen „Bye“ – Knopf)

Per Mausklick werden jeweils die einzelnen Schritte aktiviert. Vom Prinzip her wird an der Stelle die jeweilige main – Funktion des zugehörigen Teilprogramms aufgerufen und ausgeführt.

Innerhalb des Schritts „Rendern der Filme“ wird das Schnittprogramm VirtualDub über einen Kommandozeilenbefehl aufgerufen. Das Besondere hierbei ist, dass VirtualDub hierbei nicht mehr per Hand bedient werden muss. Die Einzelschritte, die beim Rendern eines Filmes erforderlich sind, erfolgen vollautomatisch. Der Aufruf von VirtualDub über die Kommandozeile erlaubt die Angabe von Parametern wie Laden der Karte für Grundeinstellungen und Pfadangabe eines Ordners, in dem die benötigten AviSynth – Skripte untergebracht sind. Dieser Ordner wird dann von VirtualDub sequentiell abgearbeitet, ähnlich wie bei einer Liste von Arbeitsaufträgen (zuvor musste diese Aufgabenliste per Hand in VirtualDub erstellt werden).
Nach Fertigstellung der Filme wird VirtualDub auch eigenständig wieder geschlossen.

Beim Herunterfahren des Rechners über das Operatingtool (hier wird der entsprechende Windows – Befehl ausgeführt) wurde eine Verzögerung von 30 Sekunden zwischen Mausklick auf dem Button und dem Start des Herunterfahrens zwischengeschaltet. Dies wird durch eine entsprechende Meldung in einem gesonderten Informationsfenster, das an der Stelle eingeblendet wird, dem Benutzer mitgeteilt. So hat man die Möglichkeit, alle noch nicht gespeicherten Arbeiten am Schnittrechner noch rechtzeitig zu sichern und, wenn erforderlich, andere noch offene Programme zu schliessen, ohne Gefahr laufen zu müssen, dass Daten beim Herunterfahren verloren gehen könnten. Das Windows – Betriebssystem bietet an dieser Stelle selbst keine Sicherheitsvorkehrungen.

Die Workflows können über die Oberfläche einzeln ausgeführt werden. Es besteht aber die Möglichkeit, einmal komplett durch alle Workflows durchzulaufen. Hier werden nacheinander alle Schritte automatisch ausgeführt, von der Erstellung der AviSynth – Skripte bis zum Herunterfahren des Rechners.

Abbildung 4 – 4 – 1 zeigt die Benutzeroberfläche des Operatingtools mitsamt den derzeit angebotenen Funktionalitäten.

Abb.4-4-1: Operatingtool

4.5 Buzzertool

Das Buzzertool wurde im Sinne des Model-View-Control-Paradigmas (MVC) programmiert. Dabei kamen zwei externe APIs zur Verwendung:
Zum einen die Bibliothek JInput.jar, welche Funktionen zur Ansteuerung der Buzzerbox zur Verfügung stellt, sowie die LwJGL.jar (Lightweight Java Game Library), eine Erweiterung zu JInput für den Zugriff auf die Hardware.

Aufgeteilt ist das Buzzertool in zwei Teilprogramme. Zum einen gibt es eine serverseitige Anwendung, die auf dem Testrechner installiert ist. Dieser Teil ist verantwortlich für den Screenshot, der von dem Monitor des Testrechners gemacht wird, und sendet das Bild als Bytedatenstrom an den Client.
Der Client ist auf dem Schnittrechner installiert und bewerkstelligt die restlichen Aufgaben des Tools.
Dazu gehören die Speicherung der empfangenen Bilder vom Server, die Sprachaufzeichnungen, das Zählen der Aufgaben, die Benutzeroberfläche inklusive Weiterverarbeitung der Eingaben des Benutzers sowie die Verarbeitung der Aktionen, die an der Buzzerbox statt finden.

Im Folgenden werden kurz die verwendeten Klassen vorgestellt und deren grundlegende Aufgabe geschildert. Die einzelnen Methoden werden ausführlich im Anhang B erläutert.

Auf der Serverseite gibt es hauptsächlich zwei Klassen:

CaptureServer

Die Klasse CaptureServer initialisiert den Server (Probandenrechner)
und stellt lediglich den Dienst, Screenshots über TCP/IP zu versenden, zur Verfügung. Die Kommunikation mit dem Client läuft entsprechend über diese Klasse ab.

ScreenShot
Die Klasse ScreenShot übernimmt das Handling der Screenshots.
und benutzt eine innere Klasse zum Zeichnen des Screenshots
in die zu speichernde Datei. Die Screenshots werden von dem Bildschirm des Rechners, auf dem der Capture-Server läuft, gemacht und in ein
Byte – Array umgewandelt. Diese Bytes werden schliesslich an das Capture-Serverobjekt weitergeleitet, das die angeforderten Daten an den Schnittrechner sendet.

Abbildung 4–5–1 zeigt einen Überblick über den Inhalt beider Klassen und stellt die Beziehung zwischen den beiden Klassen dar. Daraus geht hervor, dass das Screenshot – Objekt und das Server – Objekt voneinander abhängig sind, denn wie bereits angesprochen ist das Server – Objekt nicht für die Erstellung des Screenshots verantwortlich.

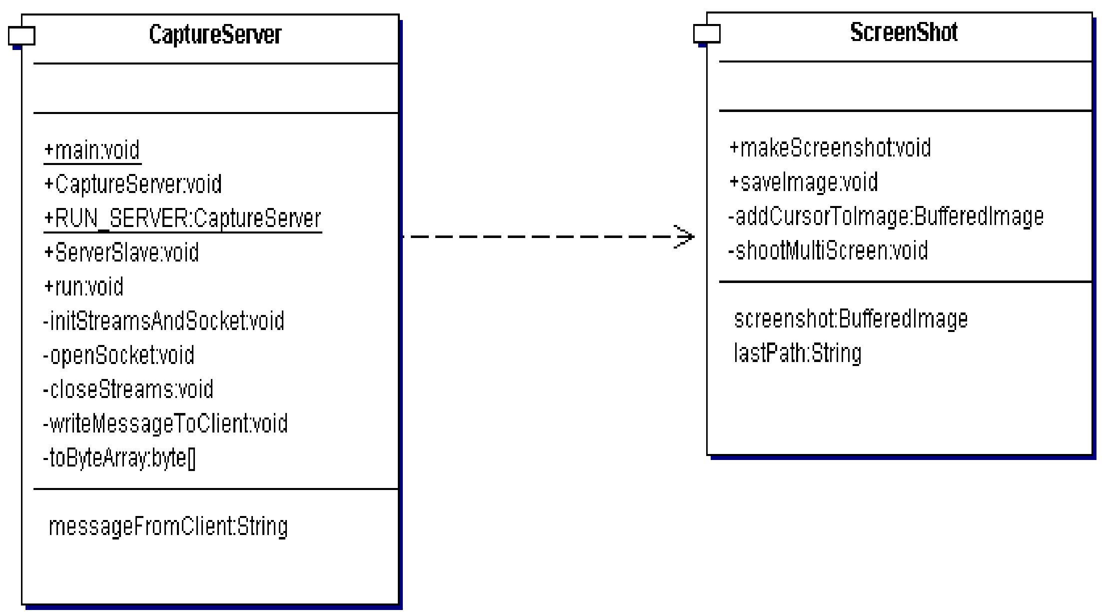


Abb.4-5-1: Buzzertool: Klassen auf der Serverseite

Auf der Client – Seite gibt es zunächst einmal folgende Hauptklassen:

CaptureClient
Die Klasse CaptureClient ist zuständig für die Anfragen an den CaptureServer.
Sie ist das Gegenstück zum CaptureServer und übernimmt die Kommunikation mit dem entsprechenden Serverobjekt. Wird ein Screenshot vom Monitor des Testrechners gewünscht, so sendet der Client ein Signal an den Server, worauf er das entsprechende Bild als Antwort vom Server zurückerhält.

PollControl
Ein Objekt der Klasse PollControl stellt den ‚Controller' der Hauptapplikation dar.
Hier werden im Sinne von MVC die erforderlichen Schritte eingeleitet, wenn irgendwelche Buzzer oder Knöpfe der Buzzerbox getätigt werden.

AudioRecorder
Die Klasse AudioRecorder übernimmt das komplette Handling der Audioaufnahmen.
Weil die Aufzeichnung auch gleichzeitig gespeichert werden soll, wird ein zusätzlicher Thread benötigt, der von einer inneren Klasse Recorder gestartet wird.

QuestionCounter
Die Klasse QuestionCounter ist zuständig für den internen Aufgabenzähler
und die Zusammenstellung der Informationen zum durchgeführten Test.
Von hier aus wird eine Statistik über die Bearbeitung der einzelnen Aufgaben während eines Tests in einer separaten Textdatei abgespeichert.

CaptureView
Die Klasse CaptureView stellt die eigentliche View im Sinne von MVC dar. Hier wird die gesamte Benutzeroberfläche des Clients (die GUI) definiert, und Eingaben sowie die anderen Aktionen auf der Oberfläche verwaltet. Entsprechende Änderungsmitteilungen werden an den Controller weitergeleitet bzw. Ausgaben, die von Aktionen an der Buzzerbox abhängen, vom Controller empfangen und umgesetzt.

ImageResizer
Die Klasse ImageResizer ist eine Hilfsklasse. Sie skaliert den letzten Screenshot neu, damit dieser auf der Benutzeroberfläche angezeigt werden kann.

Die Abbildung 4–5–2 gibt einen Überblick über die Hauptklassen des Clients.

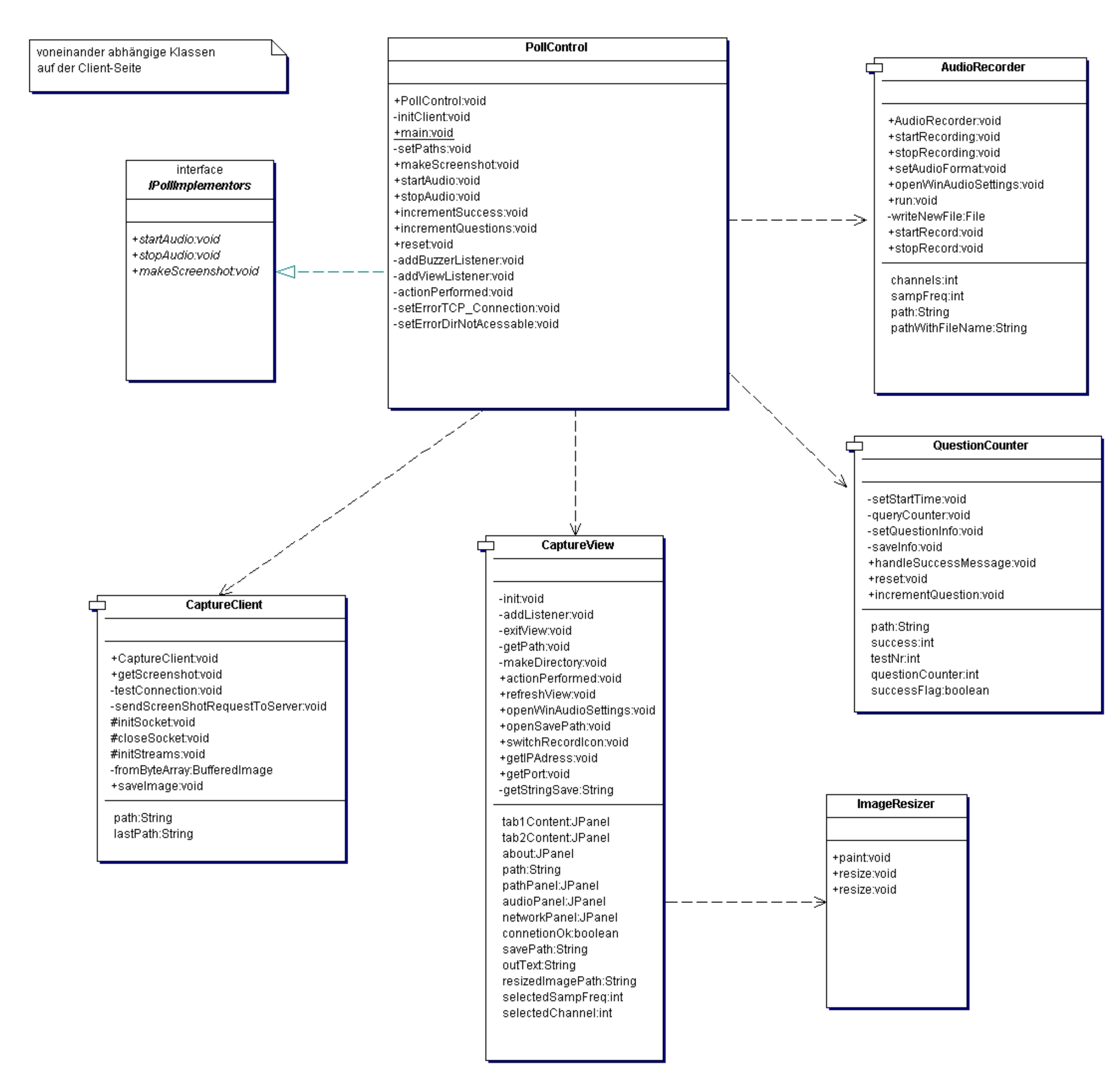


Abb. 4-5-2: Buzzertool: Klassen auf der Clientseite 1

Darüberhinaus gibt es auf der Client – Seite noch weitere nebenläufige Klassen, die verschiedene Hilfsfunktionen zur Verfügung stellen und unabhängig von den anderen Klassen sind (siehe dazu auch Abbildung 4–5–3).

PollObserver
Die Klasse PollObserver ist die Schnittstelle zur Buzzerbox, im Sinne des MVC.
Sie beinhaltet eine innere Klasse PollDaemon, die threadbasiert die Vorgänge an der Buzzerbox laufend abfragt. Der Observer dient zum "Beobachten" der Buzzerbox und teilt Veränderungen am Zustand der Buzzerbox (Knöpfe/Buzzer gedrückt oder nicht gedrückt) dem Controller – Objekt (siehe auch Klasse PollControl) der Client-Anwendung mit.

Resources
Die Klasse Resources beinhaltet lediglich wichtige Pfadangaben zu Ressourcen sowie einige Konstanten, die überall im Programm benötigt werden.

UseLabUtilities
Die Klasse UseLabUtilities stellt ein einheitliches Datums- und Zeitformat zur Verfügung und beinhaltet einige Hilfsmethoden zur Ermittlung der aktuellen Zeit.

JTextFieldFilter
Die Klasse JTextFieldFilter ist eine Hilfsklasse und überprüft die Gültigkeit einer IP-Adresse.

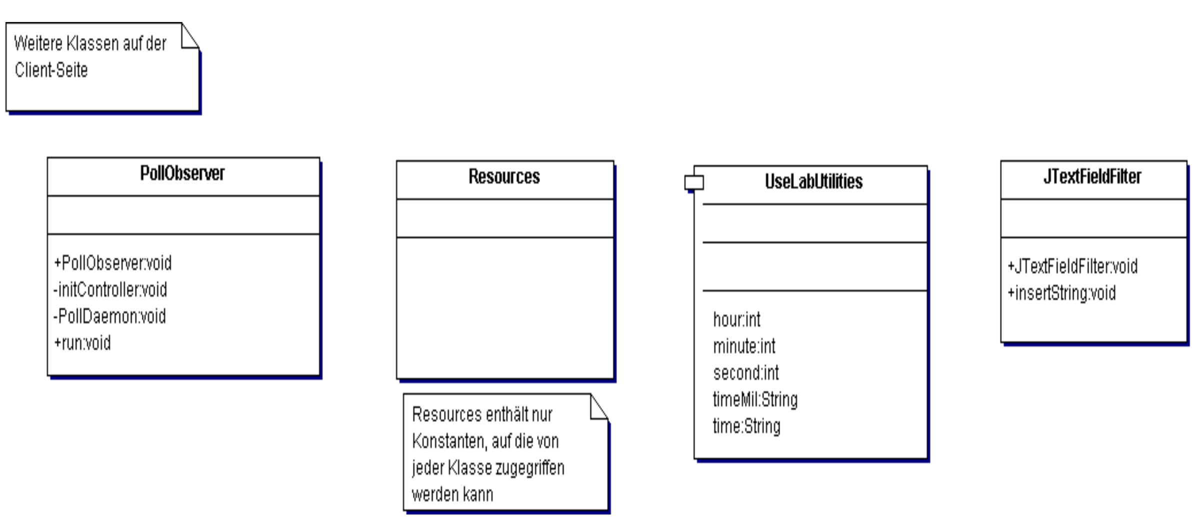


Abb. 4-5-3: Buzzertool: Klassen auf der Clientseite 2

Kapitel 5 – Erprobung

Dieses Kapitel beschreibt, wie die umgesetzten Realisierungen im laufenden Betrieb bei erstmaliger Einführung funktionierten. Ferner wird dargestellt, welche Probleme es dabei gegeben hat oder noch gibt, da nur der Einsatz in richtigen Projekten einen Einblick und eine Überprüfung auf korrekte Funktionalität gewähren und Schwachstellen und Fehler in der Soft – oder der Hardware offenbaren kann.
Hinweise auf Erweiterungen oder Verbesserungsmöglichkeiten werden an entsprechenden Stellen gegeben.

Kapitel 5.1 bildet dabei ein besonderes Kapitel, hier wird über den Einsatz und gesammelte Erfahrungen mit dem Komplettsystem berichtet.

5.1 Erprobung des neuen Gesamtsystems im Rahmen von Praxisprojekten

Die Einführung des neuen Gesamtsystems erwies sich als nicht unproblematisch. Während des Einsatzes in Praxisprojekten traten verschiedene Schwierigkeiten auf, die bis zum heutigen Stand nur teilweise gelöst werden konnten.

Zum einen besteht noch das Problem, dass die als MPEG aufgezeichneten Filme sich mit der verfügbaren Software schwer weiterverarbeiten lassen. Es gibt zwar geeignete Programme hierfür, aber diese sind zum grössten Teil unhandlich und darüberhinaus kostenintensiv. Das Problem ist zwar nicht weiter tragisch, jedoch werden dadurch zusätzliche Arbeitsschritte notwendig, die ursprünglich eigentlich vermieden werden sollten und könnten !

Die Software zur Videoaufzeichnung von der Herstellerfirma como bietet auch an, die Filme im älteren AVI – Format direkt zu speichern. Der grosse Nachteil ist jedoch, dass dann die einzelnen entstehenden Videodateien der Kameras in Bezug auf Speicherplatzbedarf explodieren. Eine Weiterverarbeitung dauert dann auch dementsprechend länger, da grössere Dateien in die Schnittsoftware geladen werden müsste.
Setzt man allerdings einen Converter ein, so werden die nach AVI umgewandelten Videodateien kleiner und handlicher. Jedoch bedeutet dies wie angesprochen mehr Arbeits – und Zeitaufwand.

Was die Geschwindigkeit, die Performanz sowie die Speicherkapazität betrifft, so hat das System eine deutliche Verbesserung erbracht. Vor allem das Zusammenschneiden der Filme ist gegenüber dem alten System deutlich schneller geworden, wenn auch nicht ganz in dem gewünschten Masse.

Ein weiteres grosses Problem ist die bislang schlechte Synchronität der Kameras bei Filmstart und Filmende. Leider werden die Kameras durch die bereitgestellte Software nacheinander verzögert gestartet und gestoppt, und nicht wie gewünscht parallel zur selben Zeit. Dies kann derzeit nicht geändert werden, hierzu ist die

Entwicklung einer neuen Software notwendig. Die Firma como ist bereits damit beauftragt, wann die verbesserte Software fertig gestellt ist, ist allerdings fraglich.

Somit muss die Synchronität der Kameras momentan manuell wiederhergestellt werden.
Möglicherweise bietet sich in diesem Zusammenhang aber auch die Ausfertigung einer weiteren Abschlussarbeit an. Eine Idee wäre es, die Filme schon während des Drehs als Live – Aufzeichnung auf einer Videoleinwand darzustellen. Dazu müsste ein eigenes Videocapturetool geschrieben werden, das die an den Videokarten eingehenden Signale abfängt, die einzelnen Videoströme zu einem einzigen Film zusammenschneidet (wie das momentan bei VirtualDub der Fall ist) und direkt als gerenderten Film wegschreibt. Entstehen würde dann bereits der fertige Film und nicht wie bisher sieben bis acht Einzelfilme, die manuell zu einem Film zusammengefügt werden müssen.

In der Anfangsphase des neuen Systems waren häufig fehlende Sequenzen in einzelnen Kameraperspektiven zu beklagen. Dies zeichnete sich dadurch aus, dass auf dem Rohmaterial Sprünge in den Videos deutlich sichtbar waren. Offenbar wurden vereinzelt Frameblöcke nicht richtig von den Capturerechnern verarbeitet und nicht mit gespeichert. Die Vermutung liegt nahe, dass das auf den beiden Rechnern installierte Betriebssystem mit der Menge an gleichzeitig ankommenden Daten überfordert ist und die einzelnen Prozesse nicht eine ausreichende Priorität und jeweils eine zu geringe Zeit für das Schreiben der Daten zugesprochen bekommen.
Man muss bedenken, dass pro Rechner von vier Videokarten permanent über einen längeren Zeitraum parallel Daten empfangen werden. Möglicherweise behindern sich somit die Prozesse gegenseitig, während sie auf die externe Festplatte zugreifen.
Ein Beweis hierfür konnte bislang nicht eindeutig erbracht werden, laut dem Task – Manager des Betriebssystems ist die Auslastung der Rechner während einer Aufzeichnung noch in einem erträglichen Rahmen.
Dieses Problem konnte durch eine Herabsetzung der Bitrate (derzeit bei
4 KB/Sekunde pro Videokarte), mit der die Bilddaten pro Sekunde weggeschrieben werden, verringert, aber nicht vollständig behoben werden. Das Problem besteht vereinzelt weiterhin, betroffen sind hauptsächlich die beiden neuen Kameras.

Insgesamt kann das neue System dennoch als eine deutliche Verbesserung gegenüber dem alten Verfahren angesehen werden.

5.2 Testen des Mousetracers

Erste Tests des neu entwickelten Mousetracers liefen bislang ohne grosse Probleme, haben jedoch an der einen oder anderen Stelle Verbesserungspotential aufgedeckt.

Zunächst einmal ist es derzeit nicht möglich, das Hintergrundbild aus den einzelnen Bildern (sowohl Mausverlauf als auch Heatmap) mit wegzuschreiben, da das Hintergrundbild (entweder das entsprechende Frame aus einem mitlaufendem Video oder ein einfaches schwarzes Bild als Kontrast zu den farbigen Mausbildern) ein eigenes Image – Objekt und nicht so einfach dem eigentlichen Mausbild untergelegt werden kann. Gespeichert wird momentan ein transparentes Bild der Heatmap bzw. des Mausverlaufs, wie in Abbildung 5 – 2 – 1 gezeigt wird.

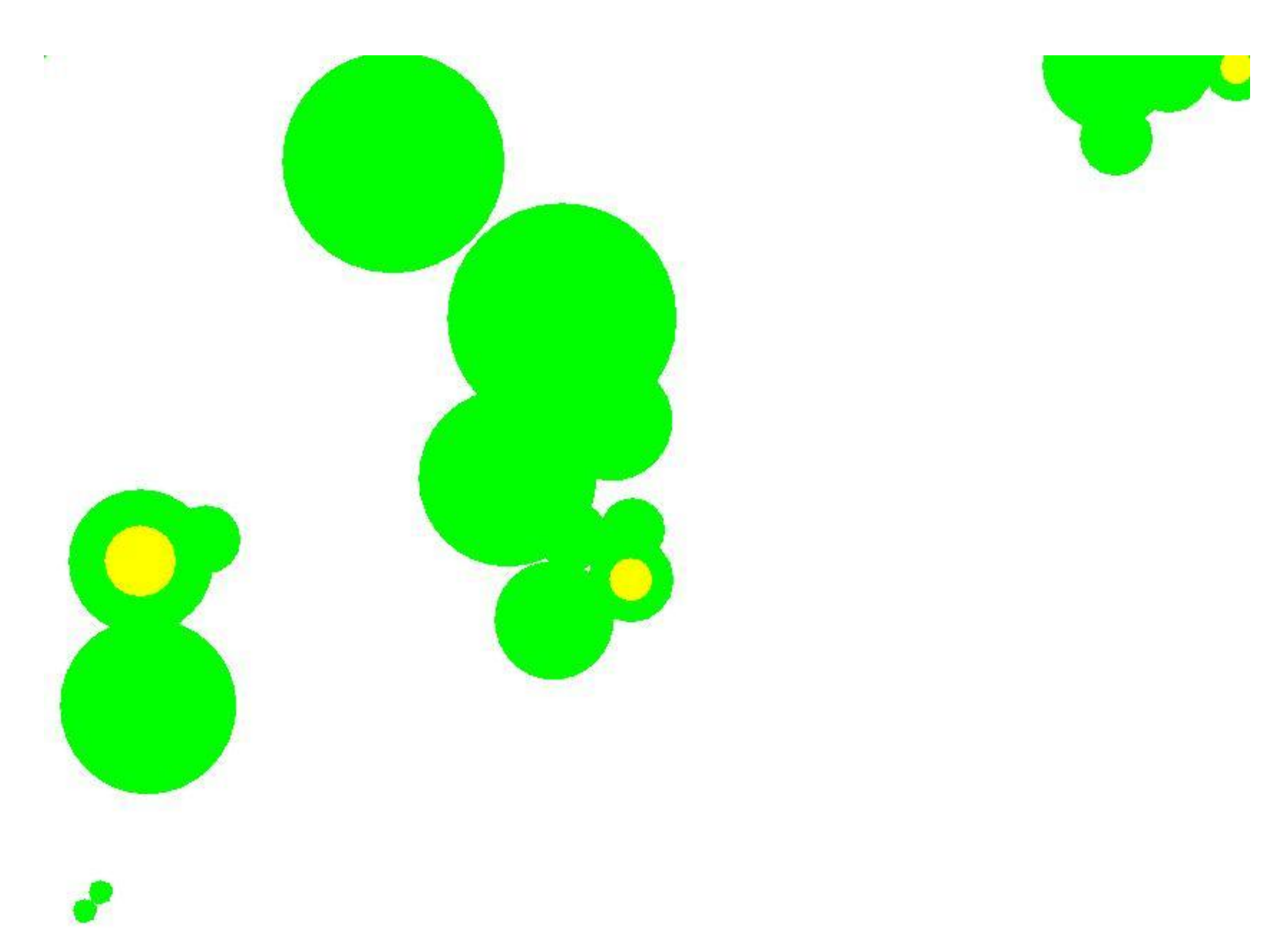


Abb. 5-2-1: Beispiel einer gespeicherten Heatmap

Will man diese Heatmap nun mit anderen Heatmaps vergelichen, so müsste man ein externes Programm verwenden, welches erlaubt, mehrere transparente Bilder übereinander zu legen.

Dennoch lässt sich an dem Bild sehr gut erkennen, welches die Hauptbereiche des auf dem Testrechner angezeigten Monitorinhalts waren, so dass dieses Bild für Vergleichszwecke mit dem Heatmap des Eyetrackers ausreichend geeignet sein sollte. Die Heatmap aus diesem Beispiel zeigt deutlich 3 Cluster, in denen sich die Mauspositionen anhäufen: Oben rechts in der Ecke am linken Rand sowie ein etwas grösserer Bereich in der Bildmitte. In diesen Bereichen wurden sämtliche Mauspositionen registriert. Die gelbe Färbung zeigt zusätzlich einen leichten Anstieg des Mausaufenthalts innerhalb der Häufungszentren.
Nicht eingefärbte Bereiche des Bildschirms wurden entsprechend nicht mit der Maus aufgesucht.

Ein weiterer Nachteil ist die Handhabung mit den beiden Slidern, um das Intervall für die Erzeugung der Heatmaps festzulegen. Ein längerer Test von etwa 30 Minuten hat beispielsweise deutlich mehr Mauspositionsdaten – bei 2 Datensätzen pro Sekunde würde dies hier insgesamt 3600 Datensätzen entsprechen. Es ist daher nachvollziehbar, dass ein Einstellen der Slider auf einen genauen Anfangs – bzw. Endpunkt mit steigender Anzahl an Datensätzen immer schwieriger wird, da für die Skala unterhalb der Slider nur ein begrenzter Platz zur Verfügung steht.

Aufgefallen ist, dass der Mousecapturer grundsätzlich als erste Mausposition zum Zeitpunkt 0 des Tests das Koordinatenpaar (0,0) ausgibt, selbst wenn der Mauszeiger nachweislich nicht auf dieser Position zu finden war. Dies ist ein Fehler innerhalb des Mousetrackers, der sich im Mousetracer weiter fortsetzt, da keine Korrekturen innerhalb der XML – Datei vorgenommen werden können und sollten.
Zu erkennen ist dies am allerersten Bild des Mausverlaufs. An dieser Stelle wird der Mauspunkt initial in der oberen linken Ecke angezeigt (aufgrund der fehlerhaften Angabe in der XML – Datei) und folglich auch mit der nächsten Mausposition verbunden, was jedoch nicht korrekt sein kann.
Dieses muss vom Testleiter bei der Auswertung unbedingt berücksichtigt werden, da das möglicherweise zu falschen Ergebnissen und Rückschlüssen führen könnte.

Weitere Tests mit dem Mousetracer konnten bis zu dem nun vorliegenden Stand noch nicht durchgeführt werden. Tests bezüglich des Zusammenspiels mit dem Eyetracker befinden sich in der Diplomarbeit von Stefan Richter [Richter 2008].

5.3 Das neue Renderverfahren

Da im Laufe des Umbaus weitere Kameras hinzugekommen waren, musste das gesamte Verfahren zur Erstellung des Videoclips immer wieder angepasst und getestet werden. Getestet wurden unter anderem die bestmögliche Ausnutzung des sichtbaren Bereichs des Clips inklusive Anordnung der Kameras, verschiedene Bildschirmauflösungen sowie die Dauer des Rendervorgangs. Auch kam es während des Einsatzes in konkreten Projekten zu Problemen, zu denen Lösungsansätze gesucht wurden. Das jetzige eingesetzte Verfahren entstand somit hauptsächlich aus den Ergebnissen des Testens.

5.3.1 Tests

Hauptaugenmerk lag in den Testdurchläufen auf dem zeitlichen Aspekt. Das Ziel soll es sein, den gesamten Vorgang zu beschleunigen. Filme wurden früher in der vierfachen Zeit der Filmlänge gerendert. Dies ist inakzeptabel geworden, besonders wenn man bedenkt, dass nicht nur ein einziger Film, sondern eine Vielzahl an Filmen in einem Durchgang gerendert werden soll.
Dabei muss jedoch beachtet werden, dass die resultierenden Videoclips in einer kleineren Auflösung vorlagen. Zudem gab es zwei Kamerafilme weniger als jetzt.

Mit Hinzunahme der beiden neuen Kameras musste die Anordnung auf dem Videoclip neu konzipiert werden. Zuvor wurden in dem unteren Drittel des Clips die vier normalen Kameraansichten in einer Reihe, und darüber der VGA – Bildschirm sowie die Aufzeichnung des Mousetrackers gezeigt.
Ein erster Versuch, die beiden neuen Kameramitschnitte mit in die Reihe der vier schon vorhandenen Kameraansichten unterzubringen, scheiterte. Für jeden der sechs Teilfilme wären in der Breite etwa nur 100 Pixel Platz verfügbar gewesen.
Die sich ergebenden Bilder waren viel zu klein, und man konnte auf denen dementsprechend kaum noch wichtige Details erkennen. Für den Auswertenden, für den gerade die beiden neuen Kameraansichten von Interesse sein dürften, da sich hier Mimik und Gestik der Testperson besonders widerspiegeln, hätte eine solche Anordnung eher Nachteile gebracht.

Die zweite Idee ist es gewesen, auf die Videoaufzeichnung des Mousetrackers zu verzichten und stattdessen den dadurch frei gewordenen Platz für die beiden neuen Kameraansichten zu verwenden. Eine vertikale Anordnung dieser beiden Kameras war dabei eine zufriedenstellende Lösung.

Leider wurde vielmals die Qualität der Filme in Bezug auf die noch niedrige Auflösung bemängelt. Daher wurde die Auflösung des Videoclips heraufgesetzt.
Hier wurden verschiedene Abmessungen getestet, was sich als eine grosse Herausforderung erwies. AviSynth verlangt eine durch vier teilbare Zahl als gültige Breite oder Höhe. Desweiteren sollte die Breite der unteren Kamerazeile optimal ausgenutzt werden. Wenn eine der vier Kameras nicht benutzt wird, so soll auf dem Videoclip keine schwarze Lücke entstehen. Dies könnte der Betrachter als störend empfinden. Stattdessen sollte eine solche Lücke in geeigneter Weise auf die drei

anderen Ansichten aufgeteilt werden. Das hat zur Folge, dass die Breite eines Kamerafilms nicht nur durch vier, sondern auch durch drei teilbar sein muss.
So entstand letztendlich nach mehreren Testversuchen die aktuelle Auflösung.

Allerdings bedeutet eine höhere Auflösung auch einen höheren Zeitaufwand, die VirtualDub beim Rendern benötigt, da jetzt wesentlich mehr Pixel pro Frame weggeschrieben werden müssen. Dies lässt die zuvor durch das neue System gewonnene Zeitersparnis wieder etwas verringern. Durch die Vorteile der höheren Auflösung bleiben die Zeiteinbußen jedoch erträglich.
Ein Test zeigt:

Dauer des Videoclips:	5 Minuten
Renderzeit auf dem alten System:	etwa 20 Minuten
Renderzeit auf dem neuen System (niedrige Auflösung):	etwa 8 Minuten
Renderzeit auf dem neuen System (höhere Auflösung):	etwa 10 Minuten

Hinweis: Die Zeitangaben können variieren, je nach Anzahl der verwendeten Kameras. Sie enthalten ausserdem nicht die Zeiten für das Konvertieren der Filme (siehe Kapitel 5.3.2).

Bezüglich des Renderns konnte das System deutlich beschleunigt werden. Da jedoch gewisse Vorbedingungen eingehalten werden müssen, um Probleme beim Rendern zu vermeiden, können diese Zahlen keinen tatsächlichen Richtwert für den kompletten Vorgang der Filmerstellung wiedergeben.

5.3.2 Probleme, Lösungsansätze und Besonderheiten

Das Hauptproblem ist zunächst die Tatsache, dass das eingesetzte Schnittprogramm VirtualDub offensichtlich nicht besonders gut mit dem Videoformat MPEG umgehen kann. Aus der Dokumentation des Programmes geht zwar hervor, dass VirtualDub zwar MPEG-Filme als Eingabemedium erlaubt, dennoch wird auch dort besonders betont, dass es dabei zu Problemen und unerwünschten Nebeneffekten kommen kann. Desweiteren unterstützt VirtualDub kein Schreiben in das MPEG – Format, so dass der resultierende Videoclip nach wie vor im AVI-Format vorliegt.
Diese Probleme haben sich im Verlauf der Zeit bestätigt. So wurden Filme von einzelnen Kameras auf dem am Ende vorliegenden Videoclip in einer anderen Geschwindigkeit abgespielt als andere Kameraansichten. Die Synchronität der einzelnen Teilfilme war deshalb mitunter nicht mehr gegeben. Zudem wurde in einigen Fällen der Ton nicht richtig mitgeschnitten bzw. fehlte ganz.
Die Filme waren somit nur schwer auswertbar.
Die neue Version von VirtualDub bietet derzeit ein Plugin für das MPEG – Format an, jedoch wird die Handhabung mit den AviSynth-Skripten dadurch erschwert. Einige verwendete Befehle von AviSynth können im MPEG – Modus nicht mehr benutzt werden. Zusätzliche Parameter wie die Angabe der exakten Framerate machen die automatische Skripterstellung zudem komplizierter und unhandlich.
Aus diesem Grund wurde sich gegen die MPEG – Variante entschieden.

Der Lösungsansatz sieht momentan so aus, dass man in einem vorgeschalteten Schritt erst einmal die einzelnen Videokanäle aus dem MPEG – Format in das

AVI – Format umwandelt. Dazu wurde das Programm „Any Video Converter“ eingeführt. Dieses unterstützt die Konvertierung von und nach verschiedenen Videoformaten wie u.a. MPEG, AVI oder WMV. Sollte in den kommenden Jahren also auf das WMV – Format umgestiegen werden, so lässt sich dieses Programm auch dafür bestens einsetzen. Die Framerate kann ebenfalls verändert werden, was besonders für die exportierte Aufnahme des Eyetrackers von Bedeutung ist.
Es kann einfach und komfortabel bedient werden, und die Konvertierung wird je nach Länge der Filme recht zügig durchgeführt. Allerdings bedeutet dieser Zwischenschritt wieder mehr Aufwand, so dass auf lange Sicht gesehen eine alternative Lösung gefunden werden sollte.

Eine weitere Vorbedingung für ein erfolgreiches Rendern ist, dass die Rohfilme vor dem eigentlichen Rendern möglicherweise einmal vorgeschnitten werden müssen, da leider sämtliche Kameras wie mehrfach angesprochen nicht zu 100% synchron bei der Aufnahme gestartet werden. Eine Lösung für dieses Problem liegt derzeit noch nicht vor, da dies vom Hersteller abhängig ist.

5.4 Archivierungstests

Zum Archivierungsvorgang ist zu sagen, dass es hier im laufenden Betrieb zu keinen grösseren Schwierigkeiten gekommen ist. Die Archivierung der Filme hat auf Anhieb und ohne Störungen geklappt. Die Sicherung der Filme ist innerhalb weniger ms (abhängig von Grösse und Anzahl der Filme) abgeschlossen.
Allerdings muss beachtet werden, dass die Linkstation für die Archivierung eingeschaltet sein muss. Dies ist zwingend notwendig, da sonst nicht auf die Linkstation zugegriffen werden kann und die Sicherung mit einer entsprechenden Fehlermeldung abgebrochen wird.

5.5 Operatingtool

Das Operatingtool wurde im Hinblick auf das Zusammenspiel der einzelnen Komponenten (vgl. Kapitel 3.8) in ausreichendem Masse ausgetestet.
Die Beobachtungen hierzu sind im nachfolgenden Abschnitt festgehalten.
Da das Tool weiter ausbaufähig ist, werden in einem zweiten, gesonderten Kapitel noch Vorschläge zur Erweiterung gemacht.

5.5.1 Einsatztests

Während des Einsatzes im Verlaufe einiger Praxisprojekte zeigte sich, dass das Operatingtool tatsächlich eine grosse Hilfe und Erleichterung während der Filmaufbereitung ist. Die Tools zur Skripterstellung und Archivierung müssen nicht mehr einzeln betrieben werden, die selbsterklärende Oberfläche bietet da eine gute Abhilfe.

Auch die Bedienung des Schnittprogramms VirtualDub per Hand entfällt komplett, da dessen Jobqueue automatisch aufgebaut und die Abarbeitung der Aufträge per Programmbefehl gestartet werden kann.

Allerdings ist auch hier zu erwähnen, dass das Programm nicht weiter fortfährt, wenn es irgendwo zu einem Fehler kommen sollte. Gibt es beispielsweise Probleme bei der Übertragung auf die Linkstation (weil diese möglicherweise nicht angeschaltet wurde), so werden alle nachfolgenden Schritte nicht mehr ausgeführt, um eventuelle grössere Schäden oder Folgefehler zu vermeiden. Insofern kann ein vollständiger Durchlauf durch die Workflows in keinem Fall zu 100 % garantiert werden.
Die einzelnen Beobachtungen, die während des separaten Renderns (Kapitel 5.3.1) und Archivierens (Kapitel 5.4) gemacht wurden, gelten selbstverständlich auch hier, da die Komponenten weitestgehend unverändert übernommen wurden.

Ein zweiter Nachteil ist die Tatsache, dass im Verlaufe des Herunterfahrens nur der Schnittrechner selbst abgeschaltet wird, nicht aber das komplette System. Die beiden Capturerechner sowie die Linkstation sollten deshalb einzeln ausgemacht werden.

5.5.2 Erweiterungsmöglichkeiten

Das Operatingtool bietet eine Menge an Erweiterungsmöglichkeiten.
So ist beispielsweise die Schnittstelle für die noch zu implementierende Komponente zum Dateitransport ins Public – Verzeichnis bereits vorhanden. Die Komponente muss nur noch mit dem entsprechenden Funktionsaufruf aktiviert werden.

Weiter wäre es sicherlich von Vorteil, wenn wie in 5.5.1 angesprochen das komplette System heruntergefahren werden könnte, und nicht nur der Schnittrechner allein. Dies könnte sich als schwierig erweisen, da bisher nicht in Erfahrung gebracht werden konnte, wie bei z.B. die Linkstation durch einen einzigen Handgriff ausgeschaltet werden könnte.

Möglicherweise lassen sich noch weitere Workflows des Operators automatisieren.
Auch diese könnten hier auf einfache Weise angekoppelt werden, da das Operatingtool programmtechnisch recht schlicht gehalten ist.
Dies gilt u.a. im Hinblick auf mögliche weitere Veränderungen im Labor wie z.B. die abschliessende Konvertierung in das WMV – Format.

Angesprochen wurde bereits die Integrierung in die Benutzeroberfläche der Filmaufzeichnungssteuerung als „Bye“ – Button. Durch eine Weiterentwicklung liesse sich die Vision der idealen Benutzerschnittstelle (Abbildung 2-3-1) somit verwirklichen.

Desweiteren wäre es bzgl. der Funktionalität des Herunterfahrens wünschenswert, wenn auch gleichzeitig die beiden Capture – Rechner mit abgeschaltet werden könnten. Somit lassen sich alle drei Rechner mit einem Mausklick herunterfahren.

5.6 Tests des Buzzertools

Das Buzzertool stellt grundlegend nur eine Erweiterung für die Auswertung dar, mit denen zusätzliche Daten festgehalten werden können. Daher wird nicht erwartet, dass es bei jedem Projekt in seinem eigentlichen Zweck eingesetzt wird.
Innerhalb der letzten Praxisprojekte hat sich jedoch herausgestellt, dass von dem Buzzertool sehr wohl exzessiver Gebrauch gemacht wird. Dabei hat sich gezeigt, dass die Bedienung gewöhnungsbedürftig ist und von den Testleitern nur nach genauer Erklärung korrekt angewendet werden kann.

Besonders die Aufgabenzählung scheint kompliziert zu sein. Obwohl darauf geachtet wurde, dass die Funktionalitäten auch zu entsprechenden passenden Farbknöpfen der Box zugewiesen wurden (weiss für richtig bearbeitete Aufgaben, schwarz für alle Aufgaben, rot zum Zurücksetzen), so dass von dem Benutzer aufgrund seiner Wahrnehmung von Farben und der daraus resultierenden Intuition der allgemeinen Bedeutung von einzelnen Farben keine schwerwiegende Fehlbedienung des Tools zu erwarten war, wurden des Öfteren die schwarze und die weisse Taste vertauscht.
Manches Mal wurde auch vergessen, die entsprechende Taste nach Ablauf einer Aufgabe zu drücken. Diese Fehlerquellen führen schliesslich zu einer Verfälschung des Ergebnisses.
Offensichtlich verlangt das Buzzertool daher dem Bediener ein grösseres Mass an Konzentration ab, als es ursprünglich angenommen wurde.
Eine ausführliche Erklärung dem Testleiter gegenüber vor Start eines Testes sowie ausreichende Beschriftung der Knöpfe sind deshalb von enormer Wichtigkeit.
Ausserdem hat der Operator während der Aufzeichnung auch einen direkten Blick auf Meldungen, die auf der Oberfläche der zugehörigen Software ausgegeben werden. So könnten entsprechende Mitteilungen des Operators an den Testleiter ebenfalls weiterhelfen.
Weiterhin ist aufgefallen, dass nicht immer die Screenshots abgespeichert werden.
Es handelt sich dabei um eine recht grosse Datenmenge, die auf die Festplatte des Schnittrechners geschrieben wird. Möglicherweise liegt dieses Problem in der grossen Arbeitsleistung, die der Schnittrechner während eines Usability – Tests zu leisten hat, da sämtliche benötigte Programme von dort aus gestartet werden und somit im Hintergrund laufen.
Tritt ein Fehler auf, so können keine weiteren gelieferten Daten des Buzzertools verarbeitet werden. Die Software steht in dem Fall still und muss neu gestartet werden. Eine Überwachung der mitlaufenden Textausgaben auf der Benutzeroberfläche durch den Operator ist deshalb sicherlich von Vorteil.

Die verschiedenen Elemente der Buzzerbox werden durch das Programm per Polling abgefangen. Daraus kann das Problem entstehen, dass eine exzessive Nutzung mehrerer Knöpfe zu fast identischen Zeiten die Software überfordert. Beispiel wäre gleichzeitiges Drücken zweier Knöpfe, dann wird nur einer der beiden korrekt in seiner Funktionalität verarbeitet. Das Programm übersieht jedoch, dass zur selben Zeit noch ein weiterer Knopf betätigt wurde.
In der Praxis ist dieser Effekt jedoch bislang nicht aufgetreten.
Alles in allem ist aber zu sagen, dass das Buzzertool seine Aufgaben erfüllt und Probleme wie oben angedeutet, die ihre Ursache nicht in der falschen Bedienung haben, selten auftreten.

Kapitel 6 – Ausblick und Zukunft

Durch den Umbau und neu hinzugekommene Geräte ist das Usability – Labor leistungsfähiger geworden. Ein effizienteres Arbeiten und qualitativ bessere Auswertungsmethoden mit Hilfe verbesserter oder neuer Werkzeuge ist nun möglich.
Filme können nun in grösserem Umfang gedreht und schneller aufgearbeitet werden als bisher. Probleme, die in der Testphase des Gesamtsystems auftraten, sind in Kapitel 5.1 beschrieben.
Das Labor, so wie es jetzt existiert, kann langfristig beibehalten werden.
Dennoch ist man hier noch lange nicht am Ziel. Es gibt noch eine Vielzahl an Möglichkeiten und Vorschlägen, mit denen man die Ausstattung und die Arbeit weiter verbessern könnte. Einige Ideen sollen hier kurz dargestellt werden:

- *Umwandlung in digitale Signale*

  Derzeit werden sämtliche Kamerabilder analog aufgenommen. Dies ergibt keine optimale Qualität der Bilder und der entstehenden Videos. Wenn die Kameras durch andere ersetzt werden, die ein digitales Bild empfangen und weiterleiten können, würde man gestochen scharfe Bilder erhalten, auf denen Details noch besser erkennbar wären.

- *Fernsehbildschirm im Form eines HDTV – Gerätes*

  Der Testleiter beobachtet die Testperson durch sechs Kameramonitore, die wie eine Wand vor ihm auf dem Tisch stehen. Ein ständiger Blick auf die Monitore strengt das Auge auf Dauer an, da die Monitore ziemlich geringer Entfernung von dem Testleiter angebracht sind. Dies steht im Widerspruch zu ergonomischen Regeln, die besagen, dass ein bestimmter Abstand zwischen Auge und Monitor einzuhalten ist. Dies führt letztendlich zur Überlastung des Testleiters, er schaut von den Monitoren weg oder wird schnell müde, und verpasst möglicherweise so wichtige Begebenheiten. Ausserdem kann ein zu geringer Abstand auf Dauer zu gesundheitlichen Schäden führen.

  Aus diesen Gründen wäre es besser, in naher Zukunft die Kameramonitore gegen ein einen neueren HDTV – Fernsehbildschirm zu ersetzen. Dieser Bildschirm könnte an der Wand angebracht werden und hätte somit einen ausreichenden Abstand zum Zuschauer.

  Mit der Einführung eines HDTV – Gerätes können zudem gleich weitere Verbesserungsvorschläge umgesetzt werden. Zum einen ist HDTV der weltweit gültige und hochauflösende Videostandard, der eine optimale Bildqualität bietet. HDTV hat im Vergleich zum herkömmlichen Bild in PAL – Norm die rund fünffach höhere Zeilenauflösung, bietet beeindruckende Detailgenauigkeit, Kontraste und Farbwiedergabe und wird immer im 16:9 – Format produziert.
  Wegen der hohen Auflösung von horizontalen und vertikalen Bildpunkten (minimal 1280 x 720, maximal sogar 1920 x 1080) hätte man genug Platz, um gleich alle sechs Kamerabilder ohne Umrechnung der Auflösung und eventuell sogar das VGA – Bild auf einem einzigen Bildschirm unterzubringen und hätte so alle Perspektiven in einem Blickfeld.

Dazu müsste eine Software entwickelt werden, die die empfangenen Bilder der Kameras live und ohne Verzögerung auf dem Gerät darstellen kann. Dies kann dadurch realisiert werden, dass ein Videoplayer entwickelt wird, der es erlaubt, sechs bzw. sieben Videos synchron abspielen zu können. Mit anderen Worten muss es möglich sein, mit einem einzelnen Mausklick die einzelnen Videostreams gleichzeitig zu starten und entsprechend auch wieder zu stoppen.
Nach ersten Untersuchungen scheint dies technisch umsetzbar. Eingesetzt werden könnte beispielsweise der VLC Media Player, der Audio – und Video – Signale als Streams entgegennimmt und sofort anzeigen kann. Man könnte also versuchen, sechs bis sieben Instanzen des VLC Media Player in der zu entwickelnden Software zu integrieren (für jeden Videostream ein VLC Player), und die Aufnahme der Streams so zu koordinieren, dass mit Hilfe von Threads die VLC – Objekte parallel angesprochen werden können. Es muss nur dafür gesorgt werden, dass die Signale, die von den sieben benutzten Videoeingängen an den beiden Capturerechnern in Empfang genommen werden, direkt zu den VLC - Objekten umgeleitet werden. Die VLC – Instanzen könnten zudem durch die Software so angeordnet werden, dass sie einen Bildschirm füllen, so dass gleichzeitig alle Kameraansichten betrachtet werden können.
Der VLC Media Player ist bereits auf dem Schnittrechner verfügbar und als Standardprogramm zum Abspielen von Videos eingerichtet worden.

Da Signale bei VLC als Streams weiterverarbeitet werden, könnte man den gedanklichen Faden weiterverfolgen. Gelingt es nun, das auf dem Bildschirm angezeigte Komplettbild, welches ja wiederum ein Stream wäre, über VLC als Video auf die Festplatte zu schreiben, so hätte man gleich nach der Aufnahme bereits den kompletten fertigen Film vorliegen. Dadurch werden Arbeitsschritte wie Schneiden und Rendern vollständig eingespart, und der Testleiter könnte bereits am Ende des Tests den Videoclip mit nach Hause nehmen.

Diese Vision ist erst gegen Ende der vorliegenden Arbeit entstanden. Ihre Umsetzung ist daher nicht Gegenstand dieser Arbeit.

- *Zusätzliche Einrichtung eines Warteraumes*

Häufig kommt es vor, dass gleich mehrere Testpersonen auf einmal im Labor erscheinen, jedoch in zeitlichen Abständen nacheinander am Testsystem arbeiten werden. So hält sich oft ein Proband im Regieraum auf, während ein anderer Test noch läuft, und bekommt dort bereits Einblicke in Geschehen und Testszenarien vor seinem eigenen Test mit. Dies ist unschön, da der künftige Proband bereits vorgewarnt wird und von den Erlebnissen und Gedanken des vorherigen Probanden, die besonders auch im Nachgespräch deutlich werden, stark beeinflusst. Eine Testauswertung ist daher nicht mehr möglich, da Ergebnisse dadurch verfälscht werden.
Besser wäre es, wenn es einen Raum gäbe, in den der nächste Proband geschickt werden und dort warten könnte, bis der vorherige Test vollständig zum Abschluss gebracht wurde. Die fertige Testperson geht nach Hause, und der nächste Proband kann aus dem Wartezimmer abgeholt werden.

- *Messung von Herzfrequenz (EKG) und Gehirnströmen (EEG)*

Bisher lassen sich bestimmte Gedankengänge und Gefühle von Probanden wie Wut oder Aufregung nur anhand des Videoclips erahnen oder durch im Nachgespräch getroffene Aussagen belegen. Vermutungen können jedoch nicht als eindeutigen Nachweis genügen. In der Medizin gibt es Geräte, die grafisch einen deutlichen Hinweis auf einzelne Regungen von Menschen darstellen können und aussagekräftiger sind. Mit dem Elektroenzephalogramm (EEG) können beispielsweise bestimmte Aktivitäten des Gehirns nachgewiesen werden, das Elektrokardiogramm (EKG) wird zur Messung von Puls und Blutdruck eingesetzt. Ein hoher Puls oder hoher Blutdruck könnte so auf Gefühlszustände wie Angst oder Aufregung des Probanden hindeuten. Stünden solche Daten für die Auswertung zur Verfügung, könnten dort auch bessere Aussagen über die Gedankengänge von Probanden getroffen werden. Nicht jeder Proband gibt in dem Nachgespräch das preis, was er wirklich während des Tests gedacht hat.

Eine Anschaffung von solchen Geräten könnte dem Auswertenden dabei helfen und zudem weitere wertvolle Informationen liefern.

Kapitel 7 – Zusammenfassung

Im Rahmen dieser Bachelor – Arbeit wurde das Usability – Labor auf einen moderneren Stand gebracht und in seinen Konzepten neu strukturiert. Die Arbeitsabläufe konnten verbessert werden. Durch Automatisierung der Workflows wurde die Erstellung der Filme vereinfacht und effizienter gemacht werden.
Mit Hilfe der neuen Ausstattung ist es nun möglich, Videos schneller und von besserer Qualität zu produzieren, als es vorher der Fall gewesen ist.

In diesem Zusammenhang wurden Probleme dargestellt, die zum Teil gelöst sind, zum Teil aber auch noch gegenwärtig bestehen, deren Hauptprobleme durch technische Ursachen und Mängel begründet sind. Diese Mängel können jedoch in naher Zukunft vermindert werden. Für die Lösung dieser Probleme sind weitere Massnahmen vorgeschlagen worden, die mittelfristig umgesetzt werden können, und die eine schnellere Auswertung von Tests ermöglichen. Somit kann das Labor auch zukünftig in seiner Qualität noch weiter ausgebaut werden.

Für eine effizientere Auswertung von Tests wurden Konzepte erarbeitet, durch die ein höheres Mass an Erkenntnissen gewonnen werden kann. Mit dem Mousetracer wurde eines dieser Konzepte realisiert. Es erlaubt die Verwertung der Daten eines bereits vorhandenen Tools, das bislang keine genügend aussagekräftigen Ergebnisse lieferte. Aus diesen Ergebnissen können nun sinnvollere Interpretationen getroffen werden, und somit dürfte das neue Auswertungswerkzeug ein unschätzbarer Gewinn für das Usability – Labor sein. Es bleibt jedoch abzuwarten, inwieweit sich der Mousetracer in der Praxis bewährt.

Die anderen Konzepte wurden aus zeitlichen Gründen nicht realisiert, es sollte aber kein Hindernis sein, diese demnächst ebenfalls noch umzusetzen.

Anhang A - Glossar

Aufgabenangemessenheit
Bei der Aufgabenangemessenheit geht es um die Frage, ob die Funktionen eines Systems (z.B. Maschine, Software oder Website) dazu dienen, die Ziele zu erreichen, die ein Nutzer damit verfolgt. Ein interaktives System ist dann aufgabenangemessen, wenn es den Benutzer unterstützt, seine Arbeitsaufgabe zu erledigen, d. h. wenn Funktionalität und Dialog auf den charakteristischen Eigenschaften der Arbeitsaufgabe basieren, anstatt auf der zur Aufgabenerledigung eingesetzten Technologie.

Automatisierung
Übertragung der menschlichen Arbeit auf Automaten unter Zuhilfenahme geeigneter Werkzeuge

AVI
Audio Video Interleave, ein von Microsoft definiertesVideo-Containerformat. In einer AVI-Datei können mehrere Video-, Audio- und Text-Untertiteldatenströme vorhanden sein, die mit verschiedenen Verfahren kodiert sein können.

Capturerechner
Hier: Rechner, der Video – und Audiosignale über Videokarten empfängt und aufzeichnet. Auch als Videograbber bezeichnet.

Effizienz
Verhältnis vom Nutzen zu dem Aufwand, mit dem der Nutzen erzielt wird.
Ein effizientes Verhalten führt wie auch ein effektives Verhalten zur Erzielung einer Wirkung, hält aber darüber hinaus den dafür notwendigen Aufwand möglichst gering.

Elektroenzephalogramm
Methode der medizinischen Diagnostik zur Messung der summierten elektrischen Aktivität des Gehirns durch Aufzeichnung der Spannungsschwankungen an der Kopfoberfläche. Ursache dieser Spannungsschwankungen sind physiologische Vorgänge innerhalb einzelnerGehirnzellen, die durch ihre elektrischen Zustandsänderungen zur Informationsverarbeitung des Gehirns beitragen.

Elektrokardiogramm
Das Elektrokardiogramm (EKG) ist die Registrierung der Summe der elektrischen Aktivitäten allerHerzmuskelfasern.

Ergonomie
Wissenschaft von der Gesetzmäßigkeit menschlicher Arbeit. Der Begriff setzt sich aus den griechischen Wörtern ergon (Arbeit, Werk) und nomos (Gesetz, Regel) zusammen. Zentrales Ziel der Ergonomie ist die Schaffung geeigneter Ausführungsbedingungen für die Arbeit des Menschen und die Nutzung technischer Einrichtungen und Werkzeuge, wobei neben der menschgerechten Gestaltung des Arbeitssystem vor allem die Verbesserung der Schnittstelle zwischen Mensch und Maschine eine besondere Bedeutung besitzt.

Erwartungskonformität
Ein Anwendungssystem ist dann erwartungskonform, wenn es den Anforderungen und Erwartungen, die ein Benutzer an das System stellt, entspricht.

Eyetracker
Gerät, mit dem man Blickbewegungen aufzeichnen und analysieren kann.

Framerate
Framerate ist die Anzahl Bilder, die von einer Kamera pro Sekunde aufgenommen werden.

FTP
File Transfer Protocol, ein Netzwerkprotokoll zur Dateiübertragung.

HAW Hamburg
Sofern nicht anders bezeichnet, ist damit die Hochschule für Angewandte Wissenschaften Hamburg gemeint.

HDTV
High Definition Television. Zeichnet sich gegenüber herkömmlichem Fernsehen durch eine erhöhte vertikale, horizontale und temporale Auflösung aus.

Intention
Absicht eines Menschen, etwas zu tun.

Interaktion
Wechselseitiges aufeinander wirken von Akteuren oder Systemen.

Lernförderlichkeit
Eine Anwendung ist lernförderlich, wenn es den Benutzer beim Erlernen des Systems unterstützt und anleitet.

Metrik
Eine Metrik ist eine (meist mathematische) Funktion, die eine Software zum Zwecke des Vergleichs oder der Bewertung in einen Zahlenwert abbildet. In dieser Arbeit sind Werte wie Mausklicks oder Tastaturanschläge gemeint.

MFC
Microsoft Foundation Classes. Eine riesige Klassenbibliothek zum Erstellen von dokumentbasierten Fensteranwendungen in der Programmiersprache C++.

MPEG
Moving Picture Experts Group ist eine Gruppe von Experten, die sich mit der Standardisierung von Videokompression und den dazugehörenden Bereichen, wie Audiodatenkompression oder Containerformaten, beschäftigt.

Mousecapturer
Anwendung, die Mausmetriken wie Mausklicks misst.

Mousetracer
Anwendung, die Mausbewegungen zu Auswertungszwecken grafisch darstellt.

MVC
Model – View – Controller, bezeichnet ein Architekturmuster zur Aufteilung von Softwaresystemen in die drei Einheiten: Datenmodell (Model), Präsentation (View) und Programmsteuerung (Controller). Ziel des Musters ist ein flexibles Programmdesign, das u. a. eine spätere Änderung oder Erweiterung erleichtern und eine Wiederverwendbarkeit der einzelnen Komponenten ermöglichen soll.

Operator
Assistent, der zumeist für die technische Bedienung zuständig ist.

PAL
Phase – Alternation – Line – Verfahren zur Farbübertragung beim analogen Fernsehen.

Proband
Gleichbedeutend mit Testperson.

Public – Verzeichnis
Ein Dateiverzeichnis, das öffentlich und somit für jedermann zugänglich ist.

RAID 5
Ein RAID – System dient zur Organisation mehrerer physischer Festplatten eines Computers zu einem logischen Laufwerk, das eine höhere Datensicherheit bei Ausfall einzelner Festplatten und einen größeren Datendurchsatz erlaubt als eine physische Platte. RAID 5 bietet sowohl gesteigerten Datendurchsatz beim Lesen von Daten als auch Redundanz, und gewährleistet Datensicherheit beim Ausfall von maximal einer Platte.

Screenshot
Abspeichern oder die Ausgabe des aktuellen graphischen Bildschirminhalts als Rastergrafik.

Selbstbeschreibungsfähigkeit
Eine Anwendung ist selbstbeschreibungsfähig, wenn jeder einzelne Dialogschritt durch Rückmeldung des Systems unmittelbar verständlich ist, selbsterklärend ist oder dem Benutzer auf Anfrage erklärt wird, so dass er nicht auf zusätzliche Hilfe angewiesen ist.

Stream
Kontinuierliche Übertragung von Daten.

S – Video
S – Video bezeichnet das getrennte Übertragen von Helligkeits – und Farbinformationen mit entsprechend ausgeführten Kabel – und Steckverbindungen.

TCP/IP
TCP/IP (Transmission Control Protocol/Internet Protocol) ist ein Netzwerkprotokoll und wird wegen seiner großen Bedeutung für das Internet auch kurz nur als Internetprotokoll bezeichnet. Die Identifizierung der Rechner geschieht über IP-Adressen.

Usability
Gebrauchstauglichkeit und Bedienerfreundlichkeit von Software.

Videograbber
Siehe Capturerechner.

VirtualDub
Freie Software zum Bearbeiten und Erstellen von Videodateien unter dem Windows – Betriebssystem.

VLC
VLC Media Player, ein freier Medienspieler und Streaming – Server.

WMV
Windows Media Video, ein neuartiger Videocodec, der Kopierschutzmassnahmen unterstützt.

Workflow
Synonym für Arbeitsschritt oder Arbeitsanweisung, die auszuführen ist.

XML
Extensible Markup Language (engl. für „erweiterbare Auszeichnungssprache“). XML ist eine Auszeichnungssprache zur Darstellung hierarchisch strukturierter Daten in Form von Textdateien. XML wird u. a. für den Austausch von Daten zwischen unterschiedlichen IT-Systemen eingesetzt, speziell über das Internet.

XviD
Freier MPEG-4 – Videocodec.

Anhang B - Bedienungsanleitungen

B1 BuzzerTool

1. Installation

Um das Buzzertool nutzen zu können, müssen zunächst einmal folgende Vorarbeiten getroffen werden:

Schritt 1:

Installation des Server-Jarfiles (BuzzerServer.jar) auf dem Testrechner.
Näheres hierzu im folgenden Abschnitt.

Schritt 2:

Installation des Client-Jarfiles (BuzzerControl.jar) auf dem Testleiterrechner (Schnittrechner). Näheres im Kapitel 1.2.

Schritt 3:

Die Buzzerbox und das Mikrophon werden an den entsprechenden Schnittstellen des Testleiterrechners angeschlossen.

1.1 Server-Jar

Um das Server-Jar zu installieren, kopiert man einfach das entsprechende Jarfile (BuzzerServer.jar) in ein beliebiges Verzeichnis.

Der Server kann danach über die Kommandozeile mit folgendem Aufruf gestartet werden:

java -jar %*PATH*%BuzzerServer.jar *portnummer*

%*PATH*% bezeichnet hierbei den aktuellen Pfad, in dem sich das Jarfile befindet.

Die Portnummer ist frei wählbar, jedoch sollten Reglementierungen wie z.B. Firewall beachten werden. Zudem muss die Portnummer mit der Portnummer im Client-Jar übereinstimmen, damit eine funktionierende Verbindung zwischen Probandenrechner und Testleiterrechner über das Buzzertool aufgebaut werden kann.

1.2 Client-Jar

Um das Client-Jar installieren, kopiert man einfach das entsprechende Jarfile (BuzzerControl.jar) in ein beliebiges Verzeichnis.

Der Client kann danach über die Kommandozeile mit folgendem Aufruf gestartet werden:

java -djava.library.path=native\win32 -jar %*PATH*%BuzzerControl.jar

%*PATH*% bezeichnet hierbei den aktuellen Pfad, in dem sich das Jarfile befindet.

Damit die Anwendung gestartet werden kann, müssen folgende Änderungen durchgeführt werden:

1. Den Explorer öffnen
2. Im Menü auf Extras→Ordneroptionen→Dateitypen gehen, den Dateityp ‚JAR' markieren und Button Erweitert anklicken
3. Aktion ‚open' auswählen und auf Bearbeiten klicken
4. im Feld Anwendung für diesen Vorgang folgenden String einfügen: "%*JAVA_HOME%*\bin\javaw.exe" **"%2"** -jar "%1" %*

Jetzt ist es möglich, über eine Verknüpfung von BuzzerControl.jar die Applikation ohne Konsole zu starten.

5. Verknüpfung von BuzzerControl.jar erstellen
6. Eigenschaften der Verknüpfung auswählen und unter Ziel folgendes einfügen:

   "*%PFAD%*\BuzzerControl.jar" -Djava.library.path=native\win32

Wobei *%PFAD%* der Pfad zum Verzeichnis ist, in dem sich BuzzerControl.jar befindet.

Weitere Installationen sind nicht notwendig.

2. Bedienungsanleitung

Es folgt nun eine ausführliche Gebrauchsanweisung des Buzzertools.

2.1 Aufbau der Buzzerbox

Die Buzzerbox besteht aus zwei grossen Buzzer sowie vier kleineren, farblich unterschiedenen Knöpfen, deren Funktionalität hier kurz angedeutet werden soll.

Linker Buzzer → Mit dem Buzzer auf der linken Seite lässt sich zu jeder Zeit ein Screenshot des gerade auf dem Testleitermonitors angezeigten Bildes machen.

Rechter Buzzer → Mit dem Buzzer auf der rechten Seite kann man zu jedem Zeitpunkt eine Sprachaufzeichnung bzgl. des Tests vornehmen, wenn etwas interessantes geschieht, das sich nicht nur allein per Bild festhalten lässt. Dazu bitte klar und deutlich in das angeschlossene Mikrofon sprechen. Die Aufzeichnung läuft, solange der Buzzer gedrückt ist.

Die bisher belegten Knöpfe sollen die spätere Auswertung des Tests erleichtern. Dazu wird am Ende des jeweiligen Tests eine Textdatei mit den gewonnenen Informationen automatisch generiert.

Weisser Knopf → Mit dem weissen Knopf (ganz rechts) werden alle erfolgreich bearbeiteten Aufgaben mitgezählt.

Schwarzer Knopf → Mit dem schwarzen Knopf werden grundsätzlich alle Aufgaben mitgezählt, egal, ob sie erfolgreich oder nicht erfolgreich bearbeitet werden konnten. Über den schwarzen Knopf wird auch der aktuelle Test gestartet.

Roter Knopf → "Reset-Knopf". Setzt alle internen Aufgabenzähler auf 0 zurück und schreibt die gesammelten Daten des letzten Tests über eine Textdatei auf die Festplatte. Mit dem roten Knopf wird das Testende signalisiert.

Grüner Knopf → Der grüne Knopf ist derzeit noch unbelegt, kann aber jederzeit mit weiteren Funktionalitäten ausgestattet werden.

2.2 Starten des Servers

Als erstes muss das Server-Jar über den schon beschriebenen Kommandozeilenbefehl gestartet werden.
Nach dem Start des Server-Jars, erscheint ein Konsolenfenster, wie in Abbildung B1 -1 dargestellt:

```
C:\WINDOWS\system32\cmd.exe
G:\PO USELAB\UseLabAPP\ScreenServer>java -jar BuzzerServer.jar 1099
Capture Server Started
```

Abb.B1- 1: Starten des Buzzerservers

Somit ist der Capture-Server bereit.
Alle eventuell auftretenden Fehlermeldungen werden hier angezeigt.

Treten keine Fehler auf, kann der Client gestartet werden.

2.3 Starten des Clients

Der Client kann, wie oben schon beschrieben, direkt über die Verknüpfung gestartet werden. Nach dem Start des Client-Jars erscheint folgende Benutzerschnittstelle (Abbildung B1 - 2):

Abb. B1- 2: Starten des Buzzer-Clients

2.4 Einstellungen vornehmen

Nachdem der Client erfolgreich gestartet werden konnte, sollten als nächstes einige Einstellungen vorgenommen werden. Die Menüseite dazu wird bei Programmstart automatisch angezeigt.

Save Path:

Unter Save Path sollte ein existierendes Verzeichnis mit Schreibrechten ausgewählt werden, in dem später alle erzeugten Dateien (Screenshots, Textdateien, Audiodateien) abgelegt werden. Achtung: Ist der Pfad ungültig oder nicht mit den erforderlichen Rechten ausgestattet, können und werden keine Dateien gespeichert !

Audio:

→ ***Sampling frequency:***

Hier muss der entsprechende Aufnahmekanal für das Mikrophon ausgewählt werden (Möglichkeiten sind: 11025 kHz, 22050 kHz oder 44100 kHz). Eventuelle Lautstärkeanpassungen sind ebenfalls hier vorzunehmen. Dazu muss das Icon auf der rechten Seite angeklickt werden, welche einen Zugriff auf die Audiokonsole von Windows bietet.

Capture Server:

→ ***IP-Adress:***

Hier sollte die gültige IP-Adresse des Servers (Probandenrechner) eingegeben werden.

→ ***Port:***

Hier muss der Port des Servers eingetragen werden. Der Port **muss** mit der Nummer, die beim Starten des Servers angegeben wurde, übereinstimmen.

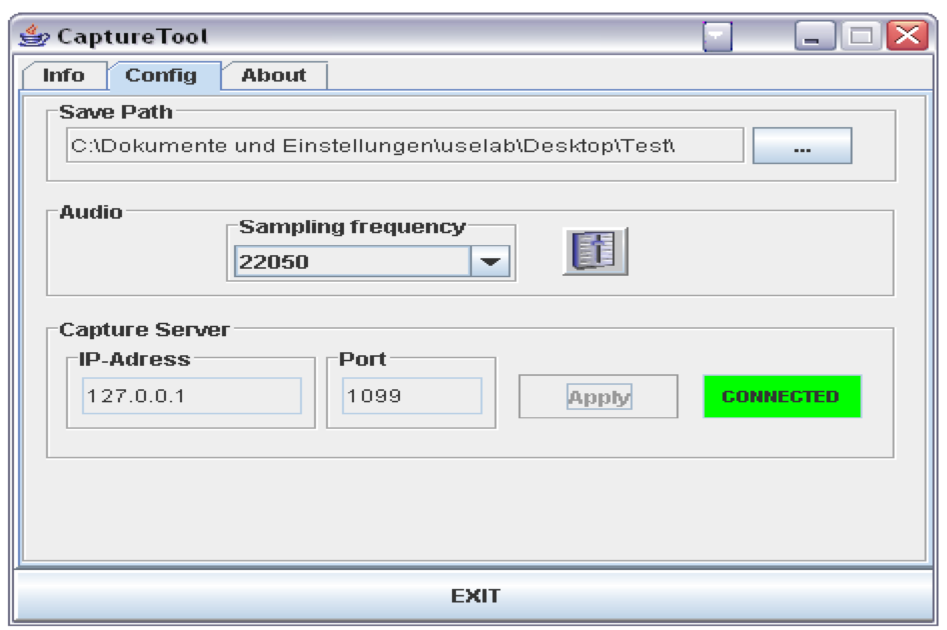


Abb. B1- 3: Der Capture-Client konnte erfolgreich mit dem Server verbunden werden

Sind alle Einstellungen vorgenommen worden, sollte mit einem Klick auf "Apply" die Verbindung zum Server hergestellt werden.

Falls der Server erreichbar ist, wechselt der Status der Anzeige (ganz rechts) auf Connected (grün), und das Buzzertool kann nun genutzt werden. Abbildung B1 – 3 zeigt einen erfolgreichen Verbindungsaufbau.

Falls der Server nicht erreichbar ist oder abstürzt, gibt es eine Fehlermeldung in der Benutzerschnittstelle, zu finden unter dem Tab "Info":

ERROR GETTING SCREENSHOT!!!
Please Check The IP And Port You Want
To Connect To!!!

Der Status des Capture-Servers wechselt von Connected auf Disconnected (rot). In diesem Fall bitte Server, IP-Adresse und Portnummer überprüfen.

Die weiteren Tabs der Client-Anwendung:

Info

Zeigt Informationen über alle Aktionen mit der Buzzerbox seit Programmstart an, wie in der nachstehenden Skizze (Abbildung B1 – 4) dargestellt.

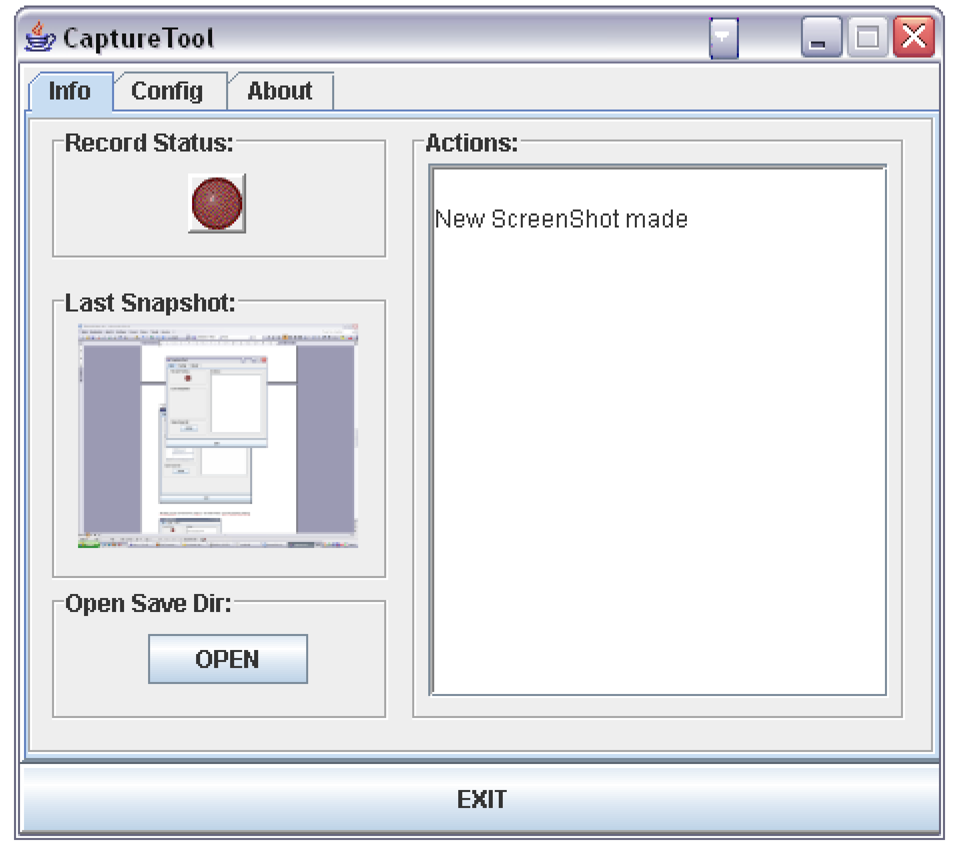


Abb. B1- 4: Das Feld Info

Record Status:

Zeigt an, ob gerade eine Sprachaufzeichnung aufgenommen wird oder nicht.

Last Screenshot:

Wenn ein Screenshot erzeugt wurde, so wird dieser hier in verkleinerter Form dargestellt. Ansonsten ist das Feld leer. Es wird immer der letzte Screenshot angezeigt.

Open Save Dir:

Mit dem Button "Open" kann man den aktuell gesetzten Speicherpfad (der unter "Config → Save Path" gesetzt wurde) öffnen und anzeigen lassen. Die Dateien in diesem Ordner sind mit Datum und Zeit betitelt.

Actions:

Einfache Textausgabe zu allen Aktionen mit der Buzzerbox seit Programmstart. Zeigt Meldungen über Beginn einer Sprachaufzeichnung, Ende einer Sprachaufzeichnung, Festhalten von Screenshots an, aber auch die aktuellen Zähler für Aufgaben gesamt und korrekt bearbeitete Aufgaben sowie eventuell auftretende Fehlermeldungen. Hier lassen sich alle Aktionen kontrollieren.
Es wird empfohlen, nach Start des Clients und den vorgenommenen Einstellungen während des Tests diesen Tab geöffnet zu halten und das Feld "Actions" im Auge zu behalten.

About

Informationsanzeigen über das Buzzertool und dessen Entwicklern.
→ Über den Button **EXIT** (zu sehen unter allen Tabs) kann der Capture-Server bzw. die Client-Applikation jederzeit beendet werden.

2.5 Festhalten eines Screenshots

Soll während des Tests ein Screenshot vom Bildschirm des Probandenrechners gemacht werden, braucht nur einmal kurz der linke Buzzer gedrückt werden.
Der Screenshot wird direkt im eingestellten Speicherpfad gespeichert und zusätzlich in der Anwendung verkleinert dargestellt. Abbildung B1 – 5 zeigt einen erfolgreich vorgenommenen Screenshot.

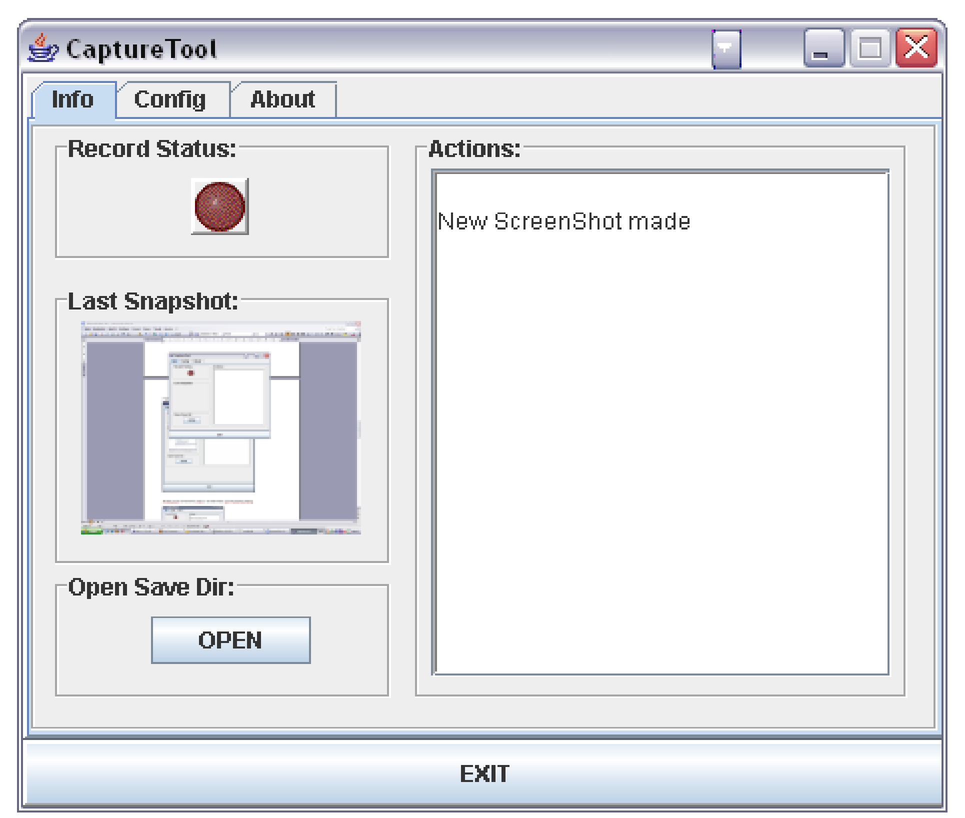


Abb. B1- 5: Screenshot war erfolgreich

Achtung !! Falls ein Zugriff auf das entsprechende Verzeichnis nicht möglich ist, erscheint folgende Fehlermeldung:

ERROR SAVING SCREENSHOT!!!
Do You Have The Rights To Access This Directory?

2.6 Festhalten einer Sprachaufzeichnung

Soll während des Tests eine Sprachaufzeichnung durchgeführt werden,
muss der rechte Buzzer getätigt werden. Während der gesamten Aufzeichnung ist der rechte Buzzer gedrückt zu halten, wird der Buzzer losgelassen, so wird auch die Aufzeichnung gestoppt !
Nun kann man seine Beobachtungen über das Mikrofon festhalten.
Die Sprachaufzeichnung wird im WAV-Format innerhalb des Speicherverzeichnisses abgespeichert (bei Fehler gibt es eine Fehlermeldung, wie beim Screenshot).

Im Feld "Record Status" unter dem Tab "Info" kann man erkennen, wann eine Sprachaufzeichnung aktiv ist. Wird gerade aufgenommen, so wird anstellen des roten Kreises ein blinkender Kreis angezeigt.

Analog zum Screenshot hier 2 Abbildungen zum Audiorecording:

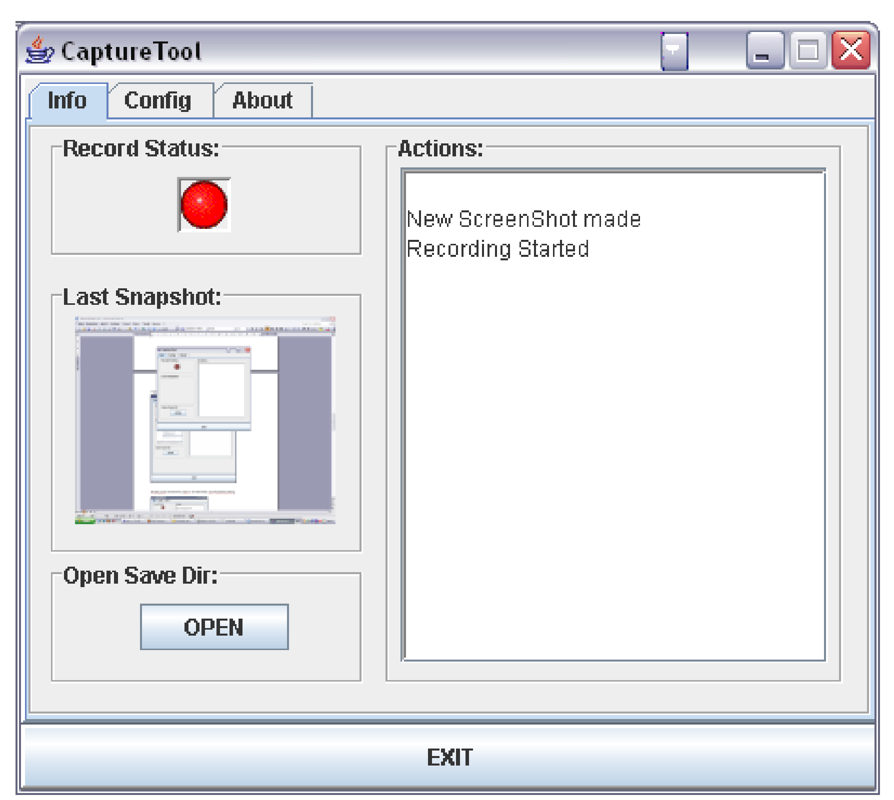


Abb. B1- 6: Eine Aufnahme wurde gestartet

Wie in Abbildung B1 – 6 dargestellt, wird durch den Eintrag „Recording Started“ im Feld Actions gemeldet, dass eine Sprachaufzeichnung gerade am Laufen ist. Abbildung B1 – 7 demonstriert entsprechend das Ende einer Aufzeichnung.

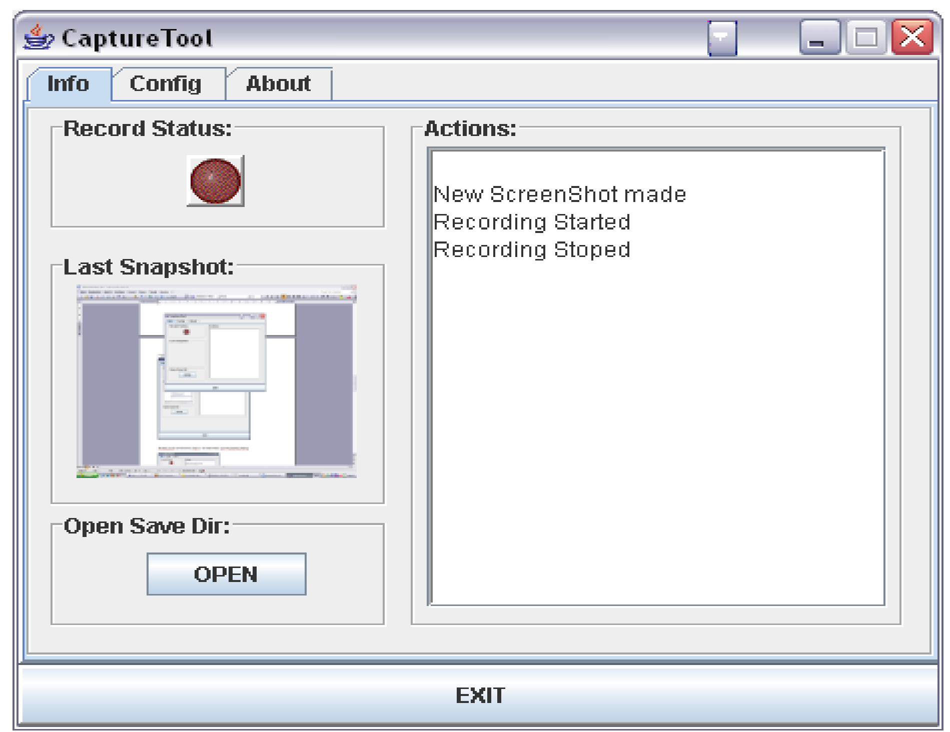


Abb. B1- 7: Die Sprachaufzeichnung wurde angehalten

2.7 Aufgabenzähler

Für den Aufgabenzähler werden drei von den vier kleineren Knöpfen der Buzzerbox benötigt: der schwarze, der weisse sowie der rote Knopf. Gezählt werden dabei die Aufgaben insgesamt (beginnend bei Aufgabe 1), die Anzahl der korrekt bearbeiteten Aufgaben (beginnend bei 1) – und damit indirekt auch die Anzahl der Aufgaben, an denen die Testpersonen gescheitert sind – sowie die benötigte Zeit für jede Aufgabe.

Achtung !
Es ist unbedingt notwendig, bei Gebrauch des Aufgabenzählers konzentriert mitzuarbeiten und wirklich nach jedem Anfang und Ende einer Aufgabe den entsprechenden Knopf zu betätigen, da sonst das Ergebnis verfälscht wird !!

Die Ergebnisse werden zum einen über die Ausgabe "Actions" unter dem Tab "Info" (betrifft die Nr. der gerade laufenden Aufgabe, ob die letzte Aufgabe erfolgreich bearbeitet wurde oder nicht, und eine Information darüber, dass der Aufgabenzähler um eine Einheit hochgezählt wurde. Auch Fehlermeldungen bezüglich des Aufgabenzählers erscheinen hier), zum anderen mit allen vollständigen Informationen in einer Textdatei abgespeichert. Die Textdatei wird allerdings erst dann im Speicherverzeichnis abgelegt, wenn mit dem **roten Knopf** signalisiert wurde, dass der laufende Test beendet wurde.

Der **rote Knopf** hat weitere wichtige Besonderheiten, siehe entsprechenden Abschnitt.

Die einzelnen Funktionalitäten im Überblick:

<u>Schwarzer Knopf:</u> Gesamtaufgaben zählen

Mit dem schwarzen Knopf werden alle Aufgaben, beginnend bei 1, gezählt. Hierbei ist es egal, ob die vorherige Aufgabe korrekt oder nicht korrekt bearbeitet wurde. Auch der interne Timer für die Dauer der jeweiligen Aufgabe wird mit dem schwarzen Knopf neu gestartet.

Es wird daher empfohlen, bei jeder neuen Aufgabe diesen Knopf zu drücken, da sonst versehentlich eine Aufgabe nicht mitgezählt wird.

Im Aktionenfeld (Tab Info → Actions) wird beim Druck auf diese Taste angezeigt, dass der Aufgabenzähler erhöht wurde und welche Aufgabe (in Form von Aufgabennummer) gestartet wurde. Man sollte diese Informationen im Auge behalten, um bei falscher Betätigung der drei Knöpfe entsprechend reagieren zu können.

<u>Weisser Knopf:</u> Korrekt bearbeitete Aufgaben zählen

Der weisse Knopf verhält sich funktional ähnlich wie der schwarze Knopf. Allerdings sollte er nur gedrückt werden, wenn eine Aufgabe erfolgreich gelöst werden konnte.

Bedingung: Der Gesamtaufgabenzähler muss zuvor um eine Einheit erhöht worden sein.
Der Zähler für die korrekt bearbeiteten Aufgaben kann nur um 1 erhöht werden, wenn der Gesamtaufgabenzähler um 1 erhöht worden ist. Gleiches gilt für die allererste Aufgabe des Tests. Der weisse Knopf kann erst dann gedrückt werden, wenn zuvor die Aufgabe 1 mit dem schwarzen Knopf als "gestartet" gesetzt worden ist.

Ansonsten wird ein entsprechender Hinweis im Aktionenfeld dargestellt, dass zuvor der schwarze Knopf zu betätigen ist.

Wird der Richtig-Aufgabenzähler erfolgreich um eins erhöht, wird der interne Timer für die Aufgabendauer gestoppt und zurückgesetzt (die Dauer der Aufgabe kann später aus der angelegten Textdatei abgelesen werden).
Ausserdem zeigt das Aktionenfeld eine entsprechende Mitteilung an, dass die Aufgabe erfolgreich gelöst werden konnte.

<u>Besonderes zwischen weisser und schwarzer Taste</u>

Wird die schwarze Taste zweimal hintereinander gedrückt, ohne zwischendurch die weisse Taste zu betätigen, so wird die aktuelle Aufgabe als "nicht gelöst" gewertet und die nächste Aufgabe gestartet !

Dies ist für die korrekte Handhabung der Tasten besonders zu beachten. Wurde also eine Aufgabe nicht gelöst, so muss keine zusätzliche Taste gedrückt werden, allerdings **darf** auch die weisse Taste dann **nicht** betätigt werden.

Die folgende Abbildung B1 – 8 zeigt die Ausgaben im Aktionenfeld, wenn

1. ein Fehler auftritt (weisse Taste gedrückt, ohne dass die Aufgabe mit schwarzer Taste freigegeben wurde)
2. eine Aufgabe korrekt erledigt wurde
3. eine Aufgabe nicht gelöst werden konnte.

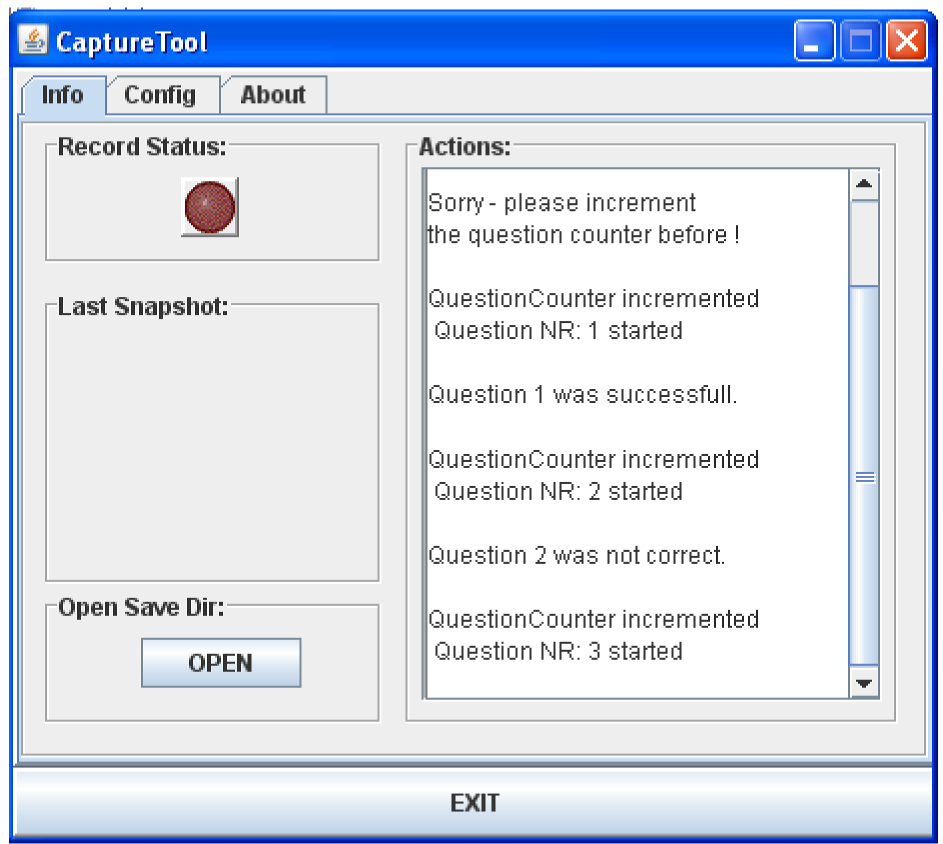


Abb. B1- 8: Schwarze und weisse Taste im Zusammenspiel – Ausgabe in der Anwendung

<u>Roter Knopf:</u> Zähler zurücksetzen und Textdatei speichern

Mit dem roten Knopf wird der laufende Test beendet. Das heisst, die Textdatei mit allen Informationen wird im Speicherverzeichnis gespeichert und die beiden Zähler wieder in den Anfangszustand zurückgesetzt. Diese Taste ist nach Ablauf der letzten Aufgabe unmittelbar zu drücken.
Ein entsprechender Hinweis ist in dem Aktionenfeld ersichtlich. Ausserdem wird dort nun eine kleine Statistik (Anzahl Aufgaben gesamt und Anzahl korrekt bearbeiteter Aufgaben) über den Test gesamt angezeigt. Ein Beispiel ist der Abbildung B1 – 9 zu entnehmen.

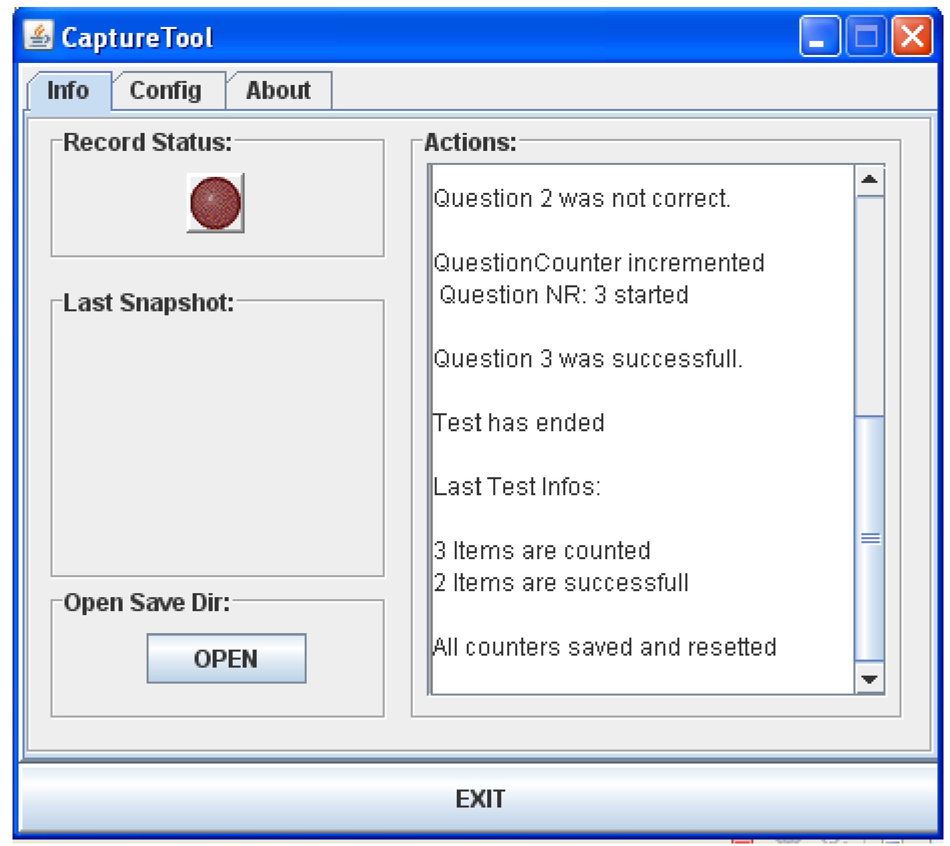


Abb. B1- 9: Die rote Taste wurde gedrückt und der aktuelle Test beendet

Im Speicherverzeichnis, welches als Speicherort eingestellt wurde, befindet sich nun ein Unterordner mit dem Namen Test+*aktuelle Testnummer* (z.B. Test1 für den ersten Testlauf seit Starten des Programms), in dem alle Dateien (Textdatei als ".txt", Sprachaufzeichnung als ".wav", Screenshots als ".png"), die zu diesem gerade abgelaufenen Test gehören, abgelegt sind. Man hat also am Ende alle zusammengehörigen Daten in einem einzigen separaten Ordner innerhalb des Speicherpfads zusammen.
Bei Betätigen des roten Knopfes wird ein weiterer interner Zähler des Buzzertools um 1 erhöht: der Testzähler. Das bedeutet, dass sofort der zweite Test beginnen kann, es wird dementsprechend für den nächsten Test ein neuer Ordner "Test2" im aktuellen Speicherverzeichnis angelegt !

Die vom Tool generierte Textdatei hat folgendes Aussehen (Abbildung B1 - 10):

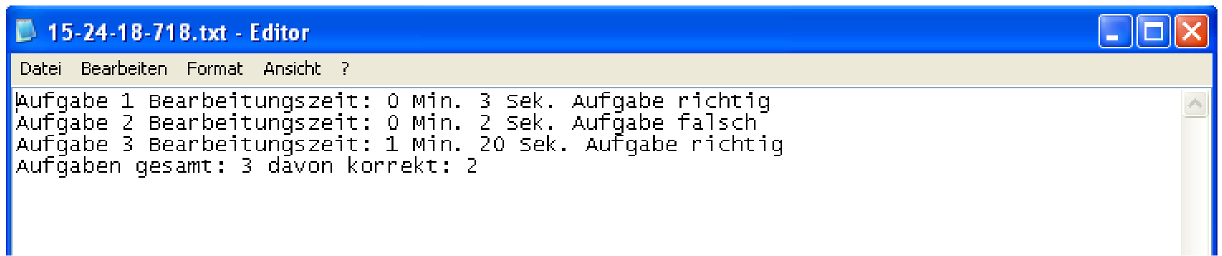


Abb. B1- 10: Die Textdatei zu dem letzten Test

Für jede Aufgabe existiert eine eigene Zeile, bestehend aus Aufgabennummer, Dauer für die Erledigung der Aufgabe sowie eine Angabe über Erfolg oder Nichterfolg bzgl. des Lösens der jeweiligen Aufgabe.
Die letzte Zeile zeigt an, wie viele Aufgaben es insgesamt im Testdurchlauf gab, und wie viele von denen erfolgreich bearbeitet werden konnten.

Diese Textdatei kann auch als Eingabe für Auswertungstools wie das von Daniel Wolf genutzt werden.

3. Hinweise zur Programmierung

Es folgt eine Übersicht über die verwendeten Klassen sowie jeweils eine kurze Beschreibung dieser Klassen und der darin enthaltenen Methoden. Hinweise für den weiteren Ausbau des Buzzertools (**grüner Knopf**) werden an den entsprechenden Stellen ebenfalls gegeben.

Die Realisierung des Buzzertools wird aus dem Grund auf diese Weise beschrieben, da hier noch Möglichkeiten zur Erweiterung vorhanden sind und diese Beschreibung gleichzeitig eine Dokumentation für die Entwickler sein soll, die sich zukünftig möglicherweise noch mit dem Tool programmtechnisch beschäftigen werden.

Package control

Das Package control stellt den Controller im Sinne des MVC dar.
Hier werden sowohl Server als auch Client gestartet.

Klassen:

CaptureServer
Die Klasse CaptureServer initialisiert den Server (Probandenrechner) und stellt den Dienst, Screenshots über TCP/IP zu versenden, zur Verfügung.
Die Screenshots werden von dem Bildschirm des Rechners, auf dem der Capture-Server läuft, gemacht.

Methoden:

public static void main(String argv[])
Startet das Programm zum Initialisieren des Servers (Aufruf von **BuzzerServer** über die Konsole und nimmt als Parameter die in der Konsole eingegebene Portnummer entgegen.

public CaptureServer(int port)
Initialisiert das Server-Objekt, ruft den Thread für den Server auf (indem ein neues ServerSlave-Objekt der inneren Klasse angelegt wird, siehe weiter unten) und initialisiert ein Screenshot-Objekt zum Testen der Verbindung zwischen Client und Server.

public static CaptureServer RUN_SERVER(int port)
Startet den Server, eine entsprechende Mitteilung wird auf der Konsole ausgegeben.

Die Klasse CaptureServer enthält eine innere Klasse ServerSlave, die für den Thread des Servers verantwortlich ist und von der JAVA-Klasse Thread abgeleitet wird. Diese innere Klasse enthält folgende Methoden:

public ServerSlave()
Startet den Server-Thread (wird vom Konstruktor der CaptureServer-Klasse aufgerufen).

public void run()
Wartet auf eine Anfrage des Clients. Erreicht eine Client-Anfrage den Server, wird ein Screenshot gemacht und zum Client zurückgesendet. Dazu muss eine Bytekonvertierung des Screenshots vorgenommen. Dieses dient nur zum Testen der Verbindung zwischen Client und Server.

private void initStreamsAndSocket()
Hier werden die nötigen Streams für den Server und den Client initialisiert. Treten Fehler auf, werden entsprechende Meldungen auf der Konsole ausgegeben.

private void openSocket()
Öffnet den Port für die Clients. Treten Fehler auf, werden entsprechende Meldungen auf der Konsole ausgegeben.

private void closeStreams()
Schliesst die Input- und Output-Streams. Treten Fehler auf, werden entsprechende Meldungen auf der Konsole ausgegeben.

private void writeMessageToClient(byte[] byteImg)
Sendet den nach Byte konvertierten Screenshot an den Client. Treten Fehler auf, werden entsprechende Meldungen auf der Konsole ausgegeben.

private String getMessageFromClient()
Methode zum Empfangen von Nachrichten, die vom Client an den Server gesendet werden. Treten Fehler auf, werden entsprechende Meldungen auf der Konsole ausgegeben.

private byte[] toByteArray(BufferedImage image)
konvertiert ein Bildobjekt (Screenshot) in ein Byte-Array.

CaptureClient
Die Klasse CaptureClient ist zuständig für die Anfragen an den CaptureServer.

Methoden:

public CaptureClient(InetAddress ip, int port)
Initialisiert das Client-Objekt und verbindet dieses mit dem Server.

public void getScreenshot()
Liest den empfangenen Screenshot, wandelt es in ein Bildobjekt zurück
und speichert das Bild. Entsprechende Methoden werden hier aufgerufen.

private void testConnection()
Testet die Client-Server-Verbindung, indem die benötigten Streams initialisiert werden und eine Anfrage nach einem Screenshot (Testbild) gesendet wird.

private void sendScreenShotRequestToServer(String mess)
Sendet eine Anfrage nach einem Screenshot an den Server. Der Server sollte daraufhin ein Byte-Array zurückliefern. Schlägt die Anfrage fehl (z.B. wenn der Server nicht erreichbar ist), wird eine entsprechende Fehlermeldung ausgegeben.

protected void initSocket(InetAddress ip, int port)
Initialisiert den nötigen Socket für die Verbindung mit dem Server.

protected void closeSocket()
Schliesst den Socket, wenn er nicht mehr benötigt wird.

protected void initStreams()
Initialisiert die nötigen Streams für den Server und den Client.
Treten Fehler auf, werden entsprechende Meldungen auf der Konsole ausgegeben.

private BufferedImage fromByteArray(byte[] imagebytes)
Konvertiert ein empfangenes Byte-Array zurück in ein Bildobjekt.

public void saveImage(BufferedImage image)
Speichert das empfangene Bild. Die Screenshots bekommen einen speziellen
Dateinamen, der sich aus aktuellen Uhrzeit (im Millisekunden) – hierfür wird
ein Hilfsobjekt der Klasse UseLabUtilities benötigt, siehe entsprechenden Abschnitt – sowie dem Typ des Bilds (".png") zusammensetzt. Gespeichert wird das Bild in dem vom Anwender eingestellten Speicherpfad.

public void setPath(String pa)
setzt den Pfad, in dem Dateien zu speichern sind.

public String getPath()
Gibt den Speicherpfad zurück.

public String getLastPath()
Gibt den gesamten Speicherpfad inklusive des kompletten Dateinamen
der zu speichernden Datei zurück.

PollControl
Ein Objekt der Klasse PollControl stellt den ‚Controller' der Hauptapplikation dar.
Hier werden im Sinne von MVC die erforderlichen Schritte eingeleitet, wenn irgendwelche Buzzer oder Knöpfe der Buzzerbox getätigt werden.
Behandlungsroutinen für den noch offenen grünen Knopf sollten ebenfalls aus dieser Klasse heraus aufgerufen werden.

Methoden:

public PollControl()
Konstruktor des Controllers. Hier werden die benötigten Objekte angelegt und initialisiert *(für den grünen Knopf die entsprechenden Objekte hier ebenfalls anlegen)*, die Listener gesetzt (wegen MVC) und Standardeinstellungen für die Benutzeroberfläche (View) bereitgestellt.

public static void main(String[] args)
Startet die Client-Anwendung.

private void initClient()
Initialisiert den Client und testet die Verbindung zum Server.

private void setPaths()
Ermittelt über den View den an der Benutzeroberfläche eingestellten Speicherpfad und gibt diese an das Recorder-, das Screenshot- und das Zähler-Objekt weiter. *Werden für den grünen Knopf weitere Objekte benötigt, sollte auch für diese der Speicherpfad in dieser Methode gesetzt werden.*

public void makeScreenshot()
Veranlasst das Screenshot-Objekt des Models dazu, einen neuen Screenshot vom Probandenrechner zu machen. Dabei wird überprüft, ob der Probandenrechner noch über das Netzwerk zu erreichen ist oder der Speicherpfad zum Speichern des Screenshots gültig ist. Ist dies nicht der Fall, werden entsprechende Fehlerbehandlungsmethoden aufgerufen.

public void startAudio()
Sendet die benötigten Angaben (unter anderem eingestellte Kanäle und Frequenzen) an das Recorder-Objekt und den View und veranlasst das Recorder-Objekt, die Sprachaufzeichnung zu starten.

public void stopAudio()
Veranlasst das Recorder-Objekt dazu, die Sprachaufzeichnung anzuhalten. Entsprechende Meldungen werden an den View weitergegeben, um das Stoppen der Aufzeichnung textlich darzustellen.

public void incrementSuccess()
Routine für den Fall, dass der weisse Schalter an der Buzzerbox getätigt wird. Wenn der Aufgabenzähler gesetzt und noch nicht der weisse Knopf gedrückt wurde, dann wird das Zählerobjekt dazu veranlasst, den Zähler für erfolgreich bearbeitete Aufgaben zu inkrementieren. Ansonsten wird an der Benutzeroberfläche eine Fehlermeldung ausgegeben.

public void incrementQuestions()
Routine für den Fall, dass der schwarze Schalter an der Buzzerbox getätigt wird. Der interne Aufgabenzähler wird über das Zähler-Objekt inkrementiert und eine entsprechende Mitteilung an der Benutzeroberfläche ausgegeben. Wenn vorher nicht der weisse Knopf gedrückt wurde, der Aufgabenzähler aber bereits gesetzt ist (>0), dann wird an der Benutzeroberfläche eine Meldung angezeigt, dass die zuletzt bearbeitete Aufgabe als nicht korrekt bewertet wird.

public void reset()
Routine für den Fall, dass der rote Schalter an der Buzzerbox getätigt wird. Veranlasst das View-Objekt dazu, eine kleine Statistik über den abgelaufenen Test an der Benutzeroberfläche auszugeben und den Speicherpfad neu zu setzen. Das Zähler-Objekt wird aufgefordert, alle seine Zähler zurückzusetzen.

private void addBuzzerListener()
Setzt einen Listener zum Abhören von Tätigkeiten, die an der Buzzerbox statt finden.

private void addViewListener()
Setzt einen Listener zum Beobachten von Vorgängen an der Benutzeroberfläche der Client-Anwendung (View).

private void actionPerformed(PropertyChangeEvent e)
Ermittelt, welche Tasten und Knöpfe der Buzzerbox gedrückt wurden und startet die damit jeweils verbundene Aktion. *Für den grünen Knopf ist eine entsprechende Behandlungsroutine hier ebenfalls aufzurufen*.

private void setErrorTCP_Connection()
Fehlerbehandlungsroutine für eine nicht vorhandene oder möglicherweise fehlerhafte Verbindung zum Server. Die Fehlermeldung wird daraufhin auf der Benutzeroberfläche angezeigt.

private void setErrorDirNotAcessable()
Fehlerbehandlungsroutine für den Fall, dass der eingestellte Speicherpfad nicht vorhanden oder mit ungenügenden Rechten ausgestattet ist. Die Meldung wird an der Benutzeroberfläche ausgegeben.

Package model

Das Package model stellt das Model im Sinne des MVC dar. Hier werden die einzelnen Buzzertool-Objekte definiert, die vom Controller benutzt werden. *Weitere Klassen für den grünen Knopf kommen ebenfalls in dieses Package.*

Klassen:

AudioRecorder
Die Klasse AudioRecorder übernimmt das komplette Handling der Audioaufnahmen. Weil die Aufzeichnung auch gleichzeitig gespeichert werden soll, wird ein zusätzlicher Thread benötigt, der von einer inneren Klasse Recorder gestartet wird.

Methoden:

public AudioRecorder()
Legt ein neues Objekt der inneren Klasse Recorder an, um den Thread nutzen zu können.

public void startRecording()
Gibt den Aufzeichnungswunsch an das Thread-Objekt weiter.
Diese Methode wird aus dem Controller heraus aufgerufen, wenn der rechte Buzzer gedrückt wird.

public void stopRecording()
Teilt dem Thread-Objekt mit, dass die Aufzeichnung beendet werden soll.
Diese Methode wird aus dem Controller heraus aufgerufen, wenn der rechte Buzzer losgelassen wird.

public void setAudioFormat(int sampFQ, int cha)
Setzt das Audioformat der Sprachaufzeichnung.

public void openWinAudioSettings()
Um weitere Einstellungen im Betriebssystem tätigen zu können,
wird über diese Methode das Windows-Programm SNDVOL32.EXE aufgerufen.

public void setChannels(int channels)
Setzt den für die Aufnahme zu verwendeten Kanal.

public void setSampFreq(int sampFreq)
Setzt die für die Aufnahme benötigte Frequenz.

public void setPath(String pa)
Setzt den Ort, an dem die Audiodatei gespeichert werden soll.

public String getPath()
Gibt den Pfad zurück, in dem die Audioaufzeichnung gespeichert wird.

Die innere Klasse Recorder für den beteiligten Thread enthält darüber hinaus folgende Methoden:

public void run()
Startet den internen Audio-Thread.

public String getPathWithFileName()
Ermittelt Speicherpfad und Dateinamen für die abzuspeichernde Audioaufzeichnung.

private File writeNewFile()
Speichert die Sprachaufzeichnung auf der Festplatte, sofern der angegebene Pfad gültig ist.

public void startRecord()
Steuert das Mikrofon an und startet die Sprachaufzeichnung. Gleichzeitig wird die Aufzeichnung durch Aufruf der Methode writeNewFile() abgespeichert. Eventuell auftretende Fehler werden abgefangen.

public void stopRecord()
Hier wird die Sprachaufzeichnung gestoppt.

ScreenShot
Die Klasse ScreenShot übernimmt das Handling der Screenshots. Auch hier wird eine innere Klasse zum Zeichnen des Screenshots in die zu speichernde Datei (dieser Vorgang läuft ebenfalls über einen Thread ab) benutzt.

Methoden:

public BufferedImage getScreenshot()
Hier wird der Screenshot erzeugt und (sofern keine Fehler auftreten) an den Controller zurückgegeben. Diese Methode wird vom Controller aufgerufen, sobald der linke Buzzer der Buzzerbox gedrückt wird.

public void makeScreenshot()
Erzeugt ein Objekt der inneren Klasse und veranlasst dieses, den Screenshot zu zeichnen. Somit wird der interne Thread gestartet.

public void saveImage(BufferedImage image, String path)
Über die Hilfsklasse UseLabUtilities wird der Dateiname für den Screenshot ermittelt und das Bild im angegebenen Pfad abgespeichert.

public String getLastPath()
Gibt den aktuell eingestellten Speicherpfad zurück.

Die innere Klasse CaptureThread für den beteiligten Thread enthält darüber hinaus folgende Methoden:

private void shootMultiScreen()
Zeichnet das Bild, das vom Server empfangen wurde, neu.

private BufferedImage addCursorToImage(BufferedImage im)
Fügt den Mauszeiger im gezeichneten Bild hinzu. Zurückgegeben wird ein gesamter neu gezeichneter Screenshot.

QuestionCounter
Die Klasse QuestionCounter ist zuständig für den internen Aufgabenzähler und die Zusammenstellung der Informationen zum durchgeführten Test.

Methoden:

public void setPath(String path)
Setzt den Pfad für die Ausgabedatei (Informationstextdatei).

private void setStartTime()
Legt über die Hilfsklasse UseLabUtilities die Anfangszeit der gerade bearbeiteten Aufgabe fest.

private void queryCounter()
Falls die Aufgabe nicht erfolgreich gelöst wurde, wird dies im String über setQuestionInfo() vermerkt. Anschliessend wird der String in einer ArrayList für die spätere Dateiausgabe zwischengespeichert. Diese Methode ist vor allem dazu da, um abzufragen, wann bei Start einer neuen Aufgabe der weisse Knopf nicht gedrückt wurde.

private void setQuestionInfo()
Ermittelt die Bearbeitungszeit für die gerade abgelaufene Aufgabe und komplettiert den String um die restlichen Informationen. Der Informationsstring ist somit folgendermassen aufgebaut:
Aufgabennummer → benötigte Zeit in Minuten:Sekunden → Erfolgreich, Ja/ Nein

private void saveInfo()
Schreibt alle Informationen aus der Arraylist (Container für die Informationsstring – 1 Informationsstring pro Aufgabe) in eine Textdatei raus.

public int getSuccess()
Liefert die Anzahl der richtig bearbeiteten Aufgaben im aktuellen Test.

public void handleSuccessMessage()
Erhöht die Anzahl der erfolgreich gelösten Aufgaben um 1 Einheit und lässt den Informationsstring über setQuestionInfo() mit den noch fehlenden Angaben komplettieren.

public void reset()
Zunächst wird hier überprüft, ob sich noch Informationen bzgl. der letzten Aufgabe im Informationsstring befinden (besonders wichtig, wenn diese nicht erfolgreich gelöst werden konnte).
Anschliessend alle internen Zähler für den nächsten Test auf den Anfangswert zurücksetzen. Der Inhalt der Arraylist (alle Informationsstrings für die bearbeiteten Aufgaben) wird über saveInfo() in eine Textdatei ausgelesen.

public int getTestNr()
Gibt die laufende Nummer des aktuellen Tests zurück (wichtig für die Erstellung des Unterordners im aktuellen Speicherpfad).

public boolean getSuccessFlag()
Gibt an, ob die letzte Aufgabe korrekt bearbeitet worden ist oder nicht.

public int getQuestionCounter()
Liefert die Anzahl aller bisherigen Aufgaben im aktuellen Test.

public void incrementQuestion()
Erhöht den Aufgabenzähler um 1, setzt den Informationsstring neu für die nächste Aufgabe und ermittelt über queryCounter() die Anfangszeit der Bearbeitung der kommenden Aufgabe. Ausserdem wird noch nach der letzten Aufgabe abgefragt, damit im Falle einer nicht korrekt gelösten Aufgabe (weisser Knopf wird dann nicht gedrückt) auch diese Information über den Fehlschlag gespeichert wird.

Package observer

Im Sinne des MVC dient das Package observer zum "Beobachten" der Buzzerbox und teilt Veränderungen am Zustand der Buzzerbox (Knöpfe/Buzzer gedrückt oder nicht gedrückt) der Client-Anwendung mit.

Klassen:

PollObserver
Die Klasse PollObserver ist die Schnittstelle zur Buzzerbox, im Sinne des MVC. Sie beinhaltet eine innere Klasse PollDaemon, die threadbasiert die Vorgänge an der Buzzerbox laufend abfragt.

Methoden:

public PollObserver()
Initialisiert ein PropertyChangeSupport-Objekt, welches dem Controller über Vorgänge an der Buzzerbox unterrichten soll.

private void initController()
Initialisiert die Buzzerbox und den Controller der Buzzerbox.

private PollDaemon()
Startet den Thread für Überwachung der Buzzerbox.

public void run()
Prüft mittels Polling nach, ob einer der grossen Buzzer oder der kleinen Schalter an der Buzzerbox gedrückt wurde und leitet entsprechende Handling-methoden ein, d.h. über das PropertyChangeSupport-Objekt wird dem Controller mitgeteilt, dass ein Ereignis an der Buzzerbox stattgefunden hat.
Die Abfrage nach dem bisher unbelegten grünen Knopf ist bereits eingebaut und braucht nicht mehr implementiert werden.

Package resources

Das Package resources beinhaltet zwei Hilfsklassen, die vom Programm
intern benötigt werden, und auf die alle anderen Klassen Zugriff haben.

Klassen:

Resources
Die Klasse Resources beinhaltet lediglich wichtige Pfadangaben zu Ressourcen sowie einige Konstanten, die überall im Programm benötigt werden. Somit sind hier keine eigenen Methoden nötig.

UseLabUtilities
Die Klasse UseLabUtilities stellt ein einheitliches Datums- und Zeitformat zur Verfügung und beinhaltet einige Hilfsmethoden zur Ermittlung der aktuellen Zeit.

Methoden:

public static int getHour()
Liefert die aktuelle Stunde.

public static int getMinute()
Liefert die aktuelle Minute.

public static int getSecond()
Liefert die gerade angebrochene Sekunde.

public static String getTime()
Liefert die Zeitangabe in einem String der Form "hh-mm-ss"

public static String getTimeMil()
Liefert die Zeitangabe in einem String der Form "hh-mm-ss-msms".
Aus diesem String wird der Dateiname der abzuspeichernden Sprachaufzeichnungen und Screenshots abgeleitet.

Package view

Das Package view stellt den View im Sinne des MVC dar.
Hier wird die gesamte Benutzeroberfläche des Clients (die GUI) definiert,
und Eingaben sowie die anderen Aktionen auf der Oberfläche verwaltet.
Entsprechende Änderungsmitteilungen werden an den Controller
weitergeleitet bzw. Ausgaben, die von Aktionen an der Buzzerbox abhängen,
vom Controller empfangen und umgesetzt.

Klassen:

CaptureView
Die Klasse CaptureView stellt die eigentliche View im Sinne von MVC dar.
Hier befindet sich die gesamte GUI der Client-Anwendung.

Methoden:

private void init()
Initialisiert die GUI mitsamt den Tabs und Buttons, legt allgemeine Position und Grösse der Benutzerschnittstelle fest, fügt Listener hinzu und nimmt allgemein gültige Einstellungen für die GUI vor.

private void addListener()
Verbindet die Buttons mit entsprechenden Listenern, die darauf horchen, wenn die Buttons mit der Maus aktiviert werden, und die damit verbundenen Aktionen/Methodenaufrufe auslösen.

private void exitView()
Blendet per Popup eine Sicherheitsabfrage ein, ob die GUI tatsächlich verlassen werden soll, und beendet im Falle eines Klicks auf den Button JA das Programm.

private JPanel getTab1Content()
Initialisiert die Inhalte des Tabs "Info" bzgl. Position, Grösse, Aussehen (hier wird auch die Grösse des abgebildeten Screenshots festgesetzt) und stattet den dort vorhandenen Button "Open" (Öffnen und Anzeigen des Speicherpfads) mit einem Listener aus.

private JPanel getTab2Content()
Verbindet die Panels für "Audio", "SavePath" und "CaptureServer" (siehe unten) zu einem Tab "Config" und legt das Layout des Tabs fest.

private JPanel getAbout()
Initialisiert die Inhalte des Tabs "About" bzgl. Position, Grösse, Aussehen.

private void getPath()
Methode zum Behandeln des Popup-Menüs, das aufgerufen wird, wenn man den Button rechts neben dem Textfeld "Save Path" auswählt – hier wird der Speicherpfad ausgewählt und gesetzt.

private void makeDirectory()
Legt im Speicherpfad einen Unterordner für die Dateien des aktuellen Tests an.

public void setPath(String path)
Setzt den Speicherpfad, in dem Textdateien, Audiodateien und Screenshots gespeichert werden sollen.

private JPanel getPathPanel()
Initialisiert die Inhalte der Fläche "SavePath" im Tab "Config" bzgl. Position, Grösse und Aussehen.

private JPanel getAudioPanel()
Initialisiert die Inhalte der Fläche "Audio" im Tab "Config" bzgl. Position, Grösse, Aussehen und stattet den dort vorhandenen Button mit einem Listener aus.

private JPanel getNetworkPanel()
Initialisiert die Inhalte der Fläche "CaptureServer" im Tab "Config"
bzgl. Position, Grösse, Aussehen und stattet den dort vorhandenen Button "Apply" (mit Server verbinden – in dem Fall werden die angegebene IP-Adresse und Portnummer auf Korrektheit überprüft) mit einem Listener aus.

public void setConnetionOk(boolean ok)
Schaltet die Anzeige von rot (disconnected) auf grün (connected),
abhängig davon, ob die Verbindung zum Server zustande gekommen ist.
Dies wird über den Parameter "ok" entschieden.

public static void main(String[] args)
Startet die Benutzeroberfläche.

public void actionPerformed(ActionEvent e)
Wird beim Betätigen des Buttons neben der Anzeige des aktuell eingestellten Speicherpfads aufgerufen und veranlasst, dass der Inhalt des Pfads angezeigt wird. Default-Einstellung ist "C:\".

public String getSavePath()
Gibt den aktuellen Speicherpfad zurück.

public void setOutText(String t)
Zeigt Textausgaben im Textfeld "Actions" an.

public void setResizedImagePath(String resizedImagePath)
Setzt den Pfad für den skalierten Screenshot.

public void refreshView(String path)
Wenn ein Screenshot gemacht wurde, so wird in dieser Methode die Skalierung des übertragenen Bildes veranlasst, damit der Screenshot auf der Benutzeroberfläche angezeigt werden kann.

public int getSelectedSampFreq()
Zeigt die ausgewählte Frequenz im entsprechenden Textfeld der GUI an.

public int getSelectedChannel()
Zeigt den ausgewählten Kanal im entsprechenden Textfeld der GUI an.

public void openWinAudioSettings()
Startet das Windows-Programm SNDVOL32.EXE zum Vornehmen von Einstellungen für die Sprachaufzeichnung.

public void openSavePath()
Öffnet den Speicherpfad direkt im Windows-Explorer.

public void switchRecordIcon()
Schaltet zwischen den Bitmaps "Tonaufnahme läuft" und "keine Tonaufnahme" um.

public void getIPAdress()
Prüft eingegebene IP-Adresse auf Gültigkeit und versucht, die Adresse über das Netzwerk zu erreichen.

public void getPort()
Nimmt die eingegebene Portnummer entgegen und überprüft sie auf Gültigkeit.

private String getStringSave(String text)
Prüft, ob eine Texteingabe des Benutzers leer ist, und wenn ja, setzt diesen String auf "0".

ImageResizer
Die Klasse ImageResizer ist eine Hilfsklasse. Sie skaliert den letzten Screenshot neu, damit dieser auf der Benutzeroberfläche angezeigt werden kann.

Methoden:

public void paint(Graphics g)
Zeichnet die Grafik.

public void resize(String path)
Liest das Bildobjekt aus dem gegebenen Pfad heraus, skaliert es auf einen konstanten Default-Wert und zeichnet es neu.

public void resize(String path, int x, int y)
Liest das Bildobjekt aus dem gegebenen Pfad heraus, skaliert es in der angegeben x- und y-Richtung und zeichnet es neu. Das Bild kann damit vergrössert oder verkleinert werden.

JTextFieldFilter
Die Klasse JTextFieldFilter ist eine Hilfsklasse und überprüft die Gültigkeit einer IP-Adresse.

Methoden:

public JTextFieldFilter(String acceptedchars)
Konstruktor. Hier wird festgelegt, welche Zeichen eine IP-Adresse beinhalten darf.

public void insertString (int offset, String str, javax.swing.text.AttributeSet attr)
Hier wird getestet, ob der übergebene String den Einstellungen bzgl. der Gültigkeit einer IP-Adresse entspricht oder nicht.

B2 VirtualDub

Virtual Dub liegt derzeit in den Versionen 1.7.5 sowie 1.7.7 als Verknüpfung auf dem Desktop des Schnittrechners bereit. Welche der beiden Versionen letztendlich benutzt wird, spielt für die Zusammenstellung der Filme keine Rolle.
Für den Fall, dass das Schneiden der Filme **nicht** automatisch über das Operatingtool erfolgen soll (sondern eigenständig), sind die nachfolgenden Schritte auszuführen.

Erstellung des Endfilms über ein AviSynth-Skript

1. VirtualDub über die Desktopverknüpfung öffnen.
2. Den Ordner mit den fertigen AviSynth-Skripten öffnen (C:\UselabOrdner\AviSynthSkripte, siehe Abbildung B2 – 1)

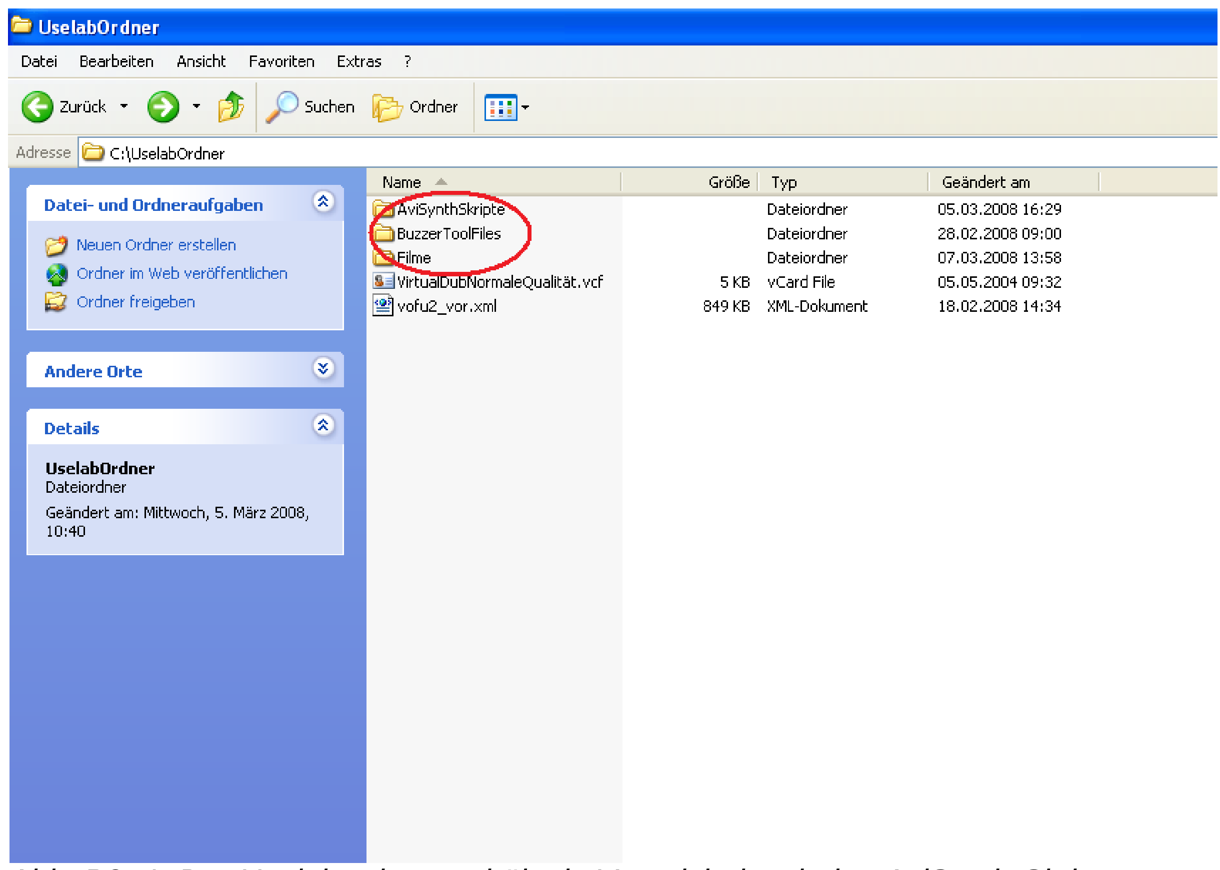


Abb. B2- 1: Der Uselabordner enthält ein Verzeichnis mit den AviSynth-Skripten

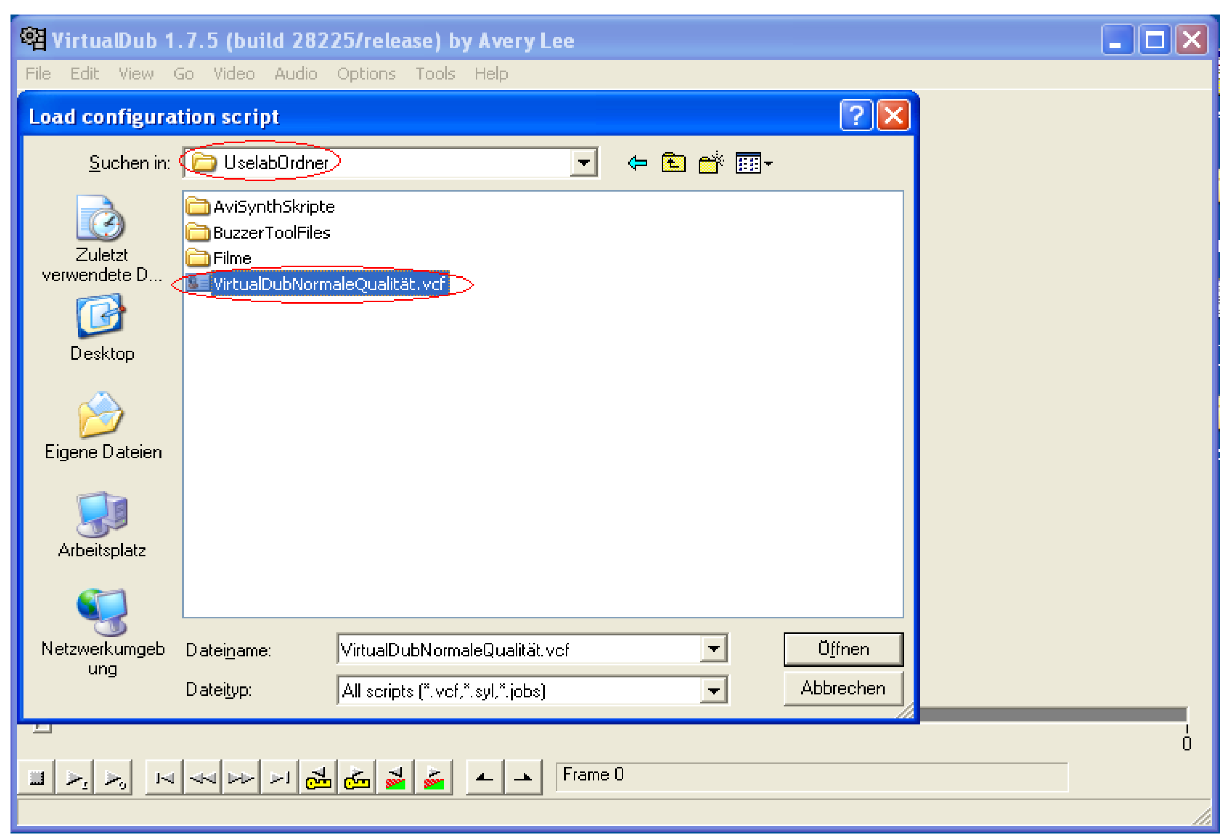


Abb. B2- 2: Karte laden

3. CTRL+L drücken und die Karte „VirtualDubNormaleQualität“ laden (diese VCF-Datei befindet sich im Uselabordner, siehe obige Abbildung B2 – 2)
4. Erstes AviSynth-Skript nach VirtualDub hineinziehen, wie in Abbildung B2 – 3 gezeigt.

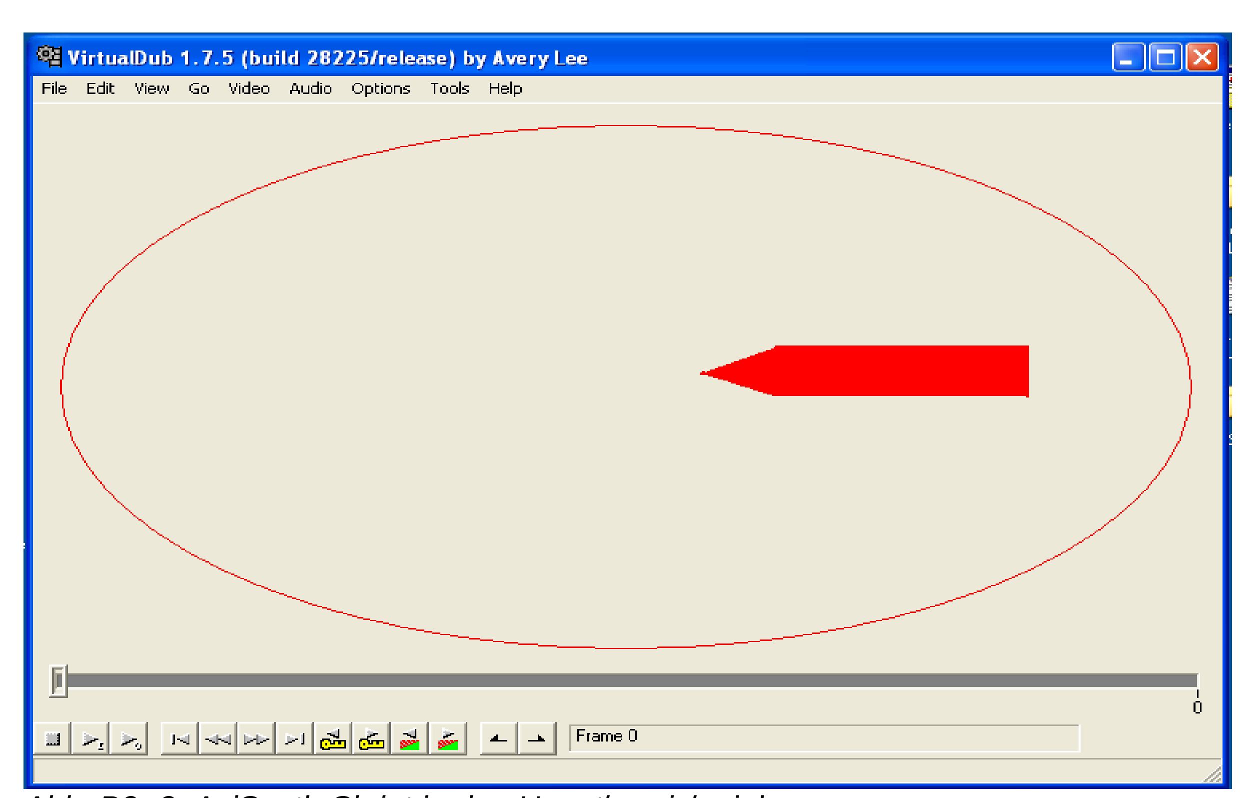


Abb. B2- 3: AviSynth-Skript in den Hauptbereich ziehen

5. Taste F7 drücken und Zielpfad des fertigen Films bestimmen. Fertige Filme sollten in das Verzeichnis C:\UselabOrdner\Filme abgelegt werden. Dem fertigen Film einen eindeutigen Namen (zum Beispiel „Eyetracker_Test_1.avi“) vergeben. Haken bei „Don’t run job now“ setzen. Siehe hierzu Abbildung B2 – 4.

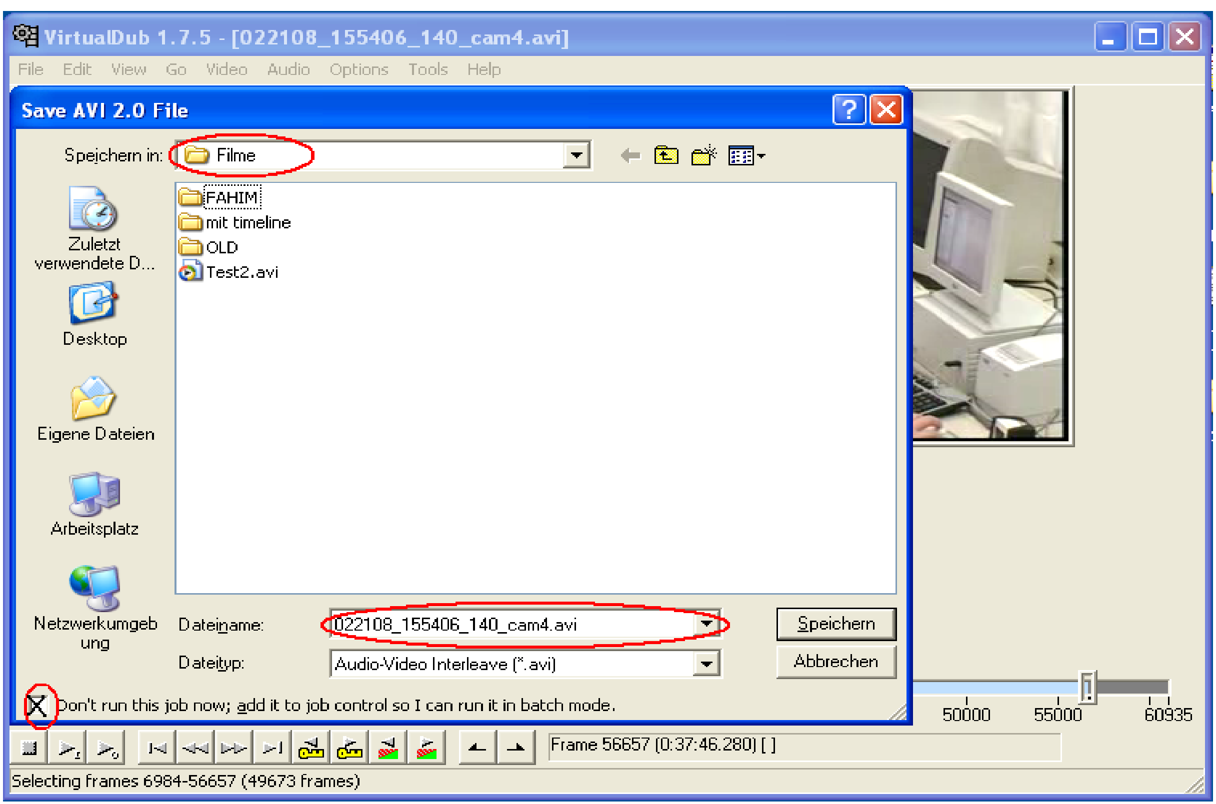


Abb. B2- 4: Einstellungen vor dem Rendern

6. Schritte 4 und 5 für weitere Skripte wiederholen (falls vorhanden).
7. Taste F4 und dann auf „Start“ drücken, um die Jobqueue abzuarbeiten (siehe Abbildung B2 – 5). Die Filme werden nacheinander automatisch fertiggestellt.

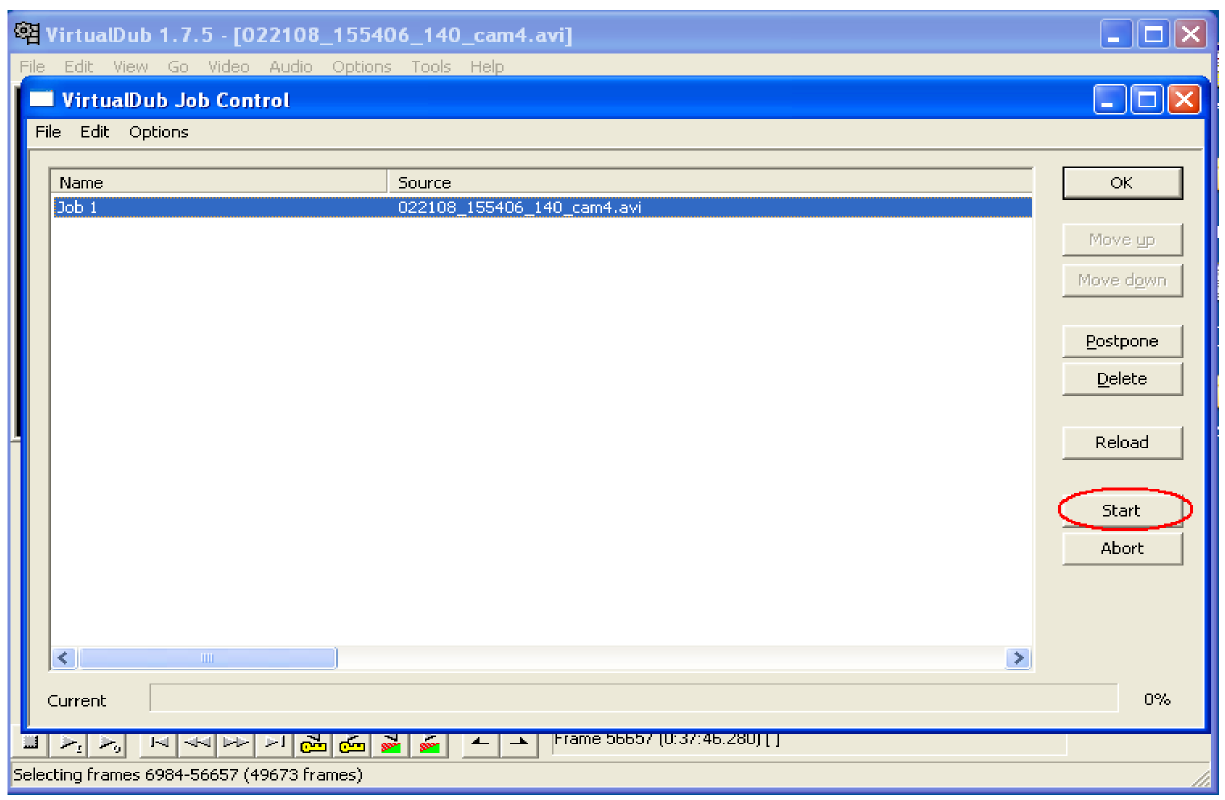


Abb. B2- 5: Job Control von VirtualDub

8. Zum Abschluss Queue leeren, indem alle Jobs aus der Liste entfernt werden.

Manuelles Synchronisieren der Einzelfilme durch Zurechtschneiden derselben

Da es unter Umständen sein kann, dass die Filmaufnahmen der einzelnen Kameras nicht vollständig synchron zueinander sind, kann VirtualDub dazu verwendet werden, eine Synchronisierung der Einzelfilme manuell durchzuführen.

1. VirtualDub über die Desktopverknüpfung öffnen.
2. Einen Rohfilm (**muss** im AVI-Format vorliegen) nach VirtualDub hineinziehen. Es ist günstig, dass auf diesem Rohfilm auch der Bildschirm des Testrechners zu sehen ist. Daher bieten sich entweder Kamera 3 oder Kamera 4 für diesen Schritt an.
3. Einen markanten Punkt (z.B. eine bestimmte Bewegung der Testperson) am Anfang des Films suchen. Hinweis: Mit dem Schieberegler kann man den Film vor – und zurückspulen lassen.

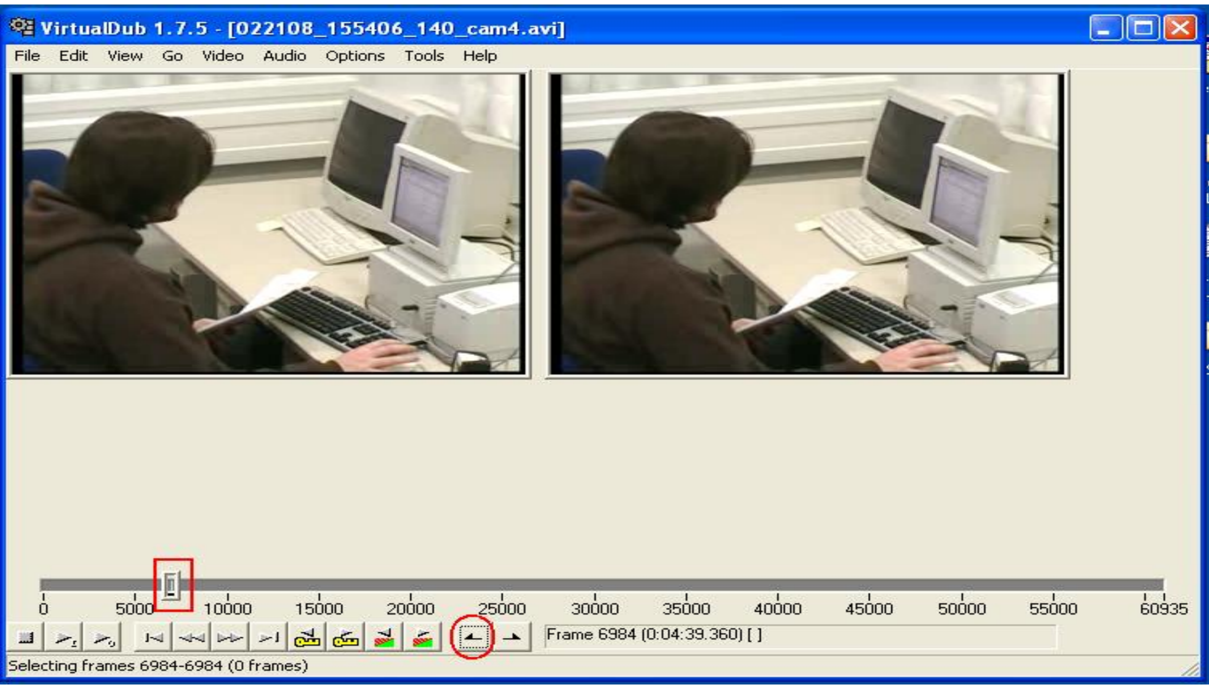


Abb. B2- 6: Anfangspunkt setzen

4. Button zum Setzen der Startposition betätigen (Abbildung B2 – 6).
5. Schritt 3 für das Ende des Films wiederholen.
6. Button zum Setzen der Endposition betätigen (Abbildung B2 – 7).

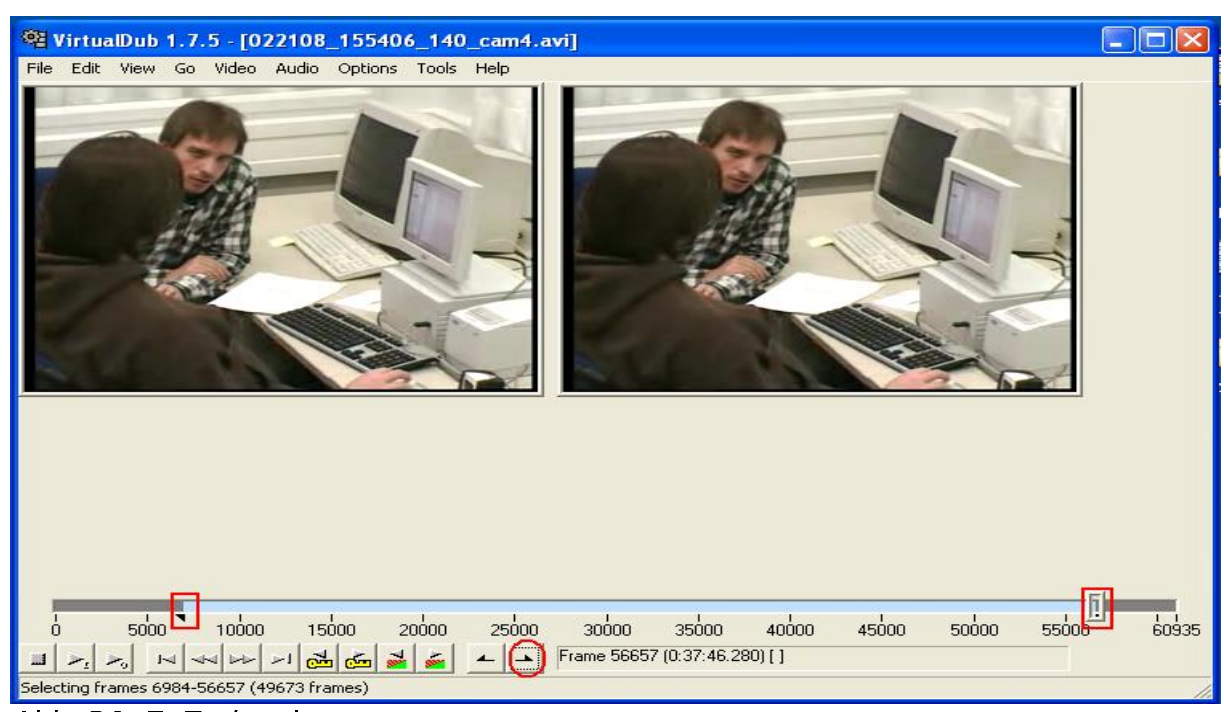


Abb. B2- 7: Endpunkt setzen

7. Taste F7 drücken und Zielpfad des geschnittenen Films bestimmen. Dieser Film sollte sich am Ende im gleichen Verzeichnis wie der originale Rohfilm befinden.
8. Schneidevorgang starten.
9. Schritte 3-8 für alle anderen Kameras wiederholen. Man sollte sich hierbei an den markanten Punkten des zuerst geschnittenen Films orientieren.

B3 Operatingtool

Das Operatingtool stellt eine Arbeitserleichterung des Operators dar. Mit ihm werden nahezu alle Arbeitsschritte des Operators automatisiert. Die einzelnen Workflows lassen sich sowohl einzeln als auch als Gesamteinheit in einem einzigen Schritt ausführen.

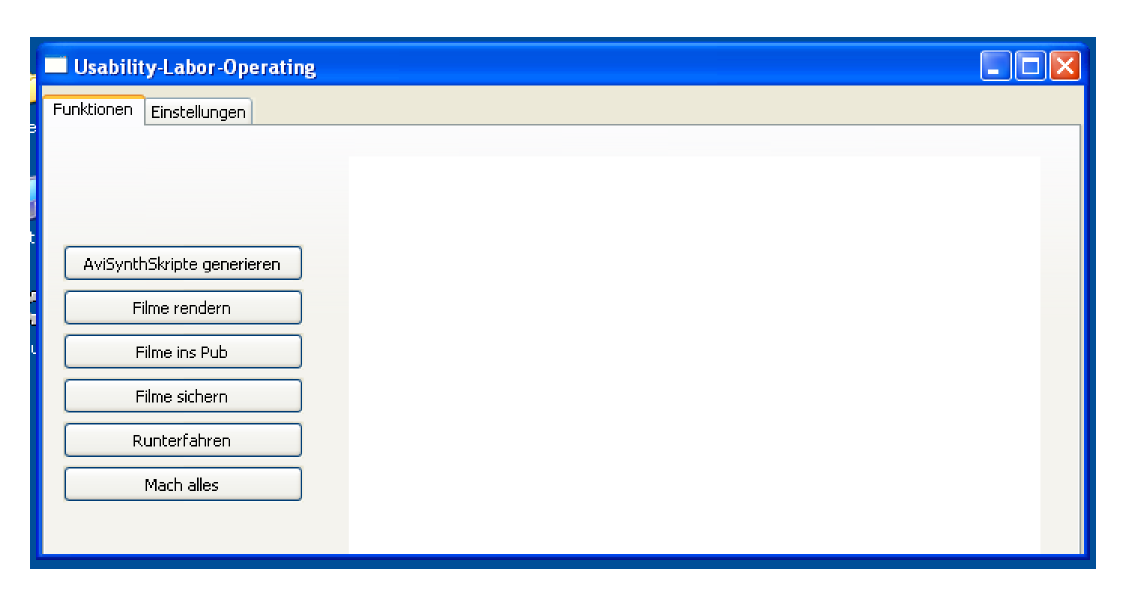


Abb. B3- 1: Funktionsübersicht des Operatingtools

Wie der Abbildung B3 – 1 entnommen werden kann, können folgende Schritte durchgeführt werden:

- Unter „AviSynthSkripte generieren“ können für alle (!) vorhandenen Rohfilme die entsprechenden AviSynth-Skripte automatisch erstellt werden. Zu beachten ist hierbei, dass gleich für jeden aufgenommenen Usability-Test das Skript erstellt wird, nicht nur für einen einzelnen Test.
  Nach Fertigstellung der Skripte wird eine entsprechende Meldung im Textfenster ausgegeben (siehe Abbildung B3 – 2).

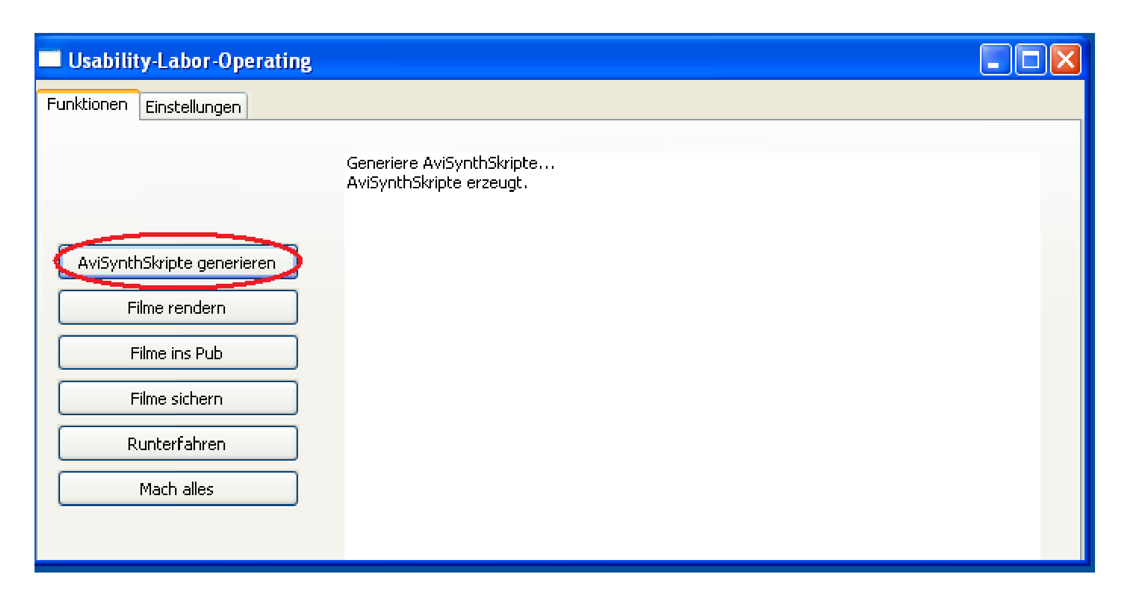


Abb. B3- 2: AviSynth-Skripte erzeugen

- Unter „Filme rendern“ können die erzeugten AviSynth-Skripte in Endfilme umgewandelt werden. Eine Bedienung des Programms VirtualDub ist nicht mehr nötig, dies geschieht vollständig automatisch (siehe Abbildung B3 – 3).

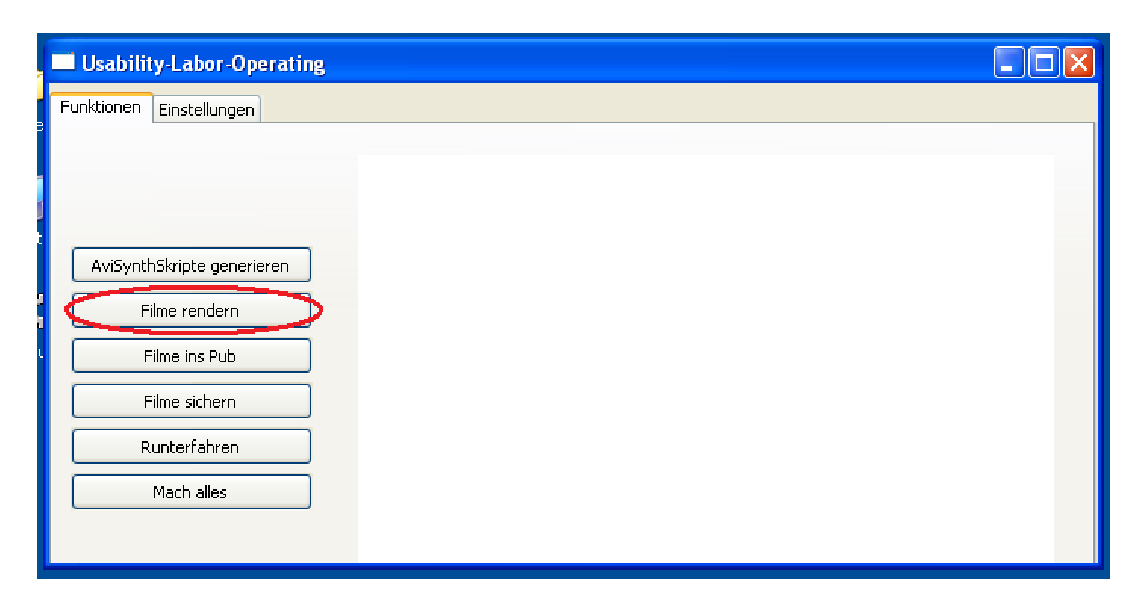


Abb. B3- 3: Filme automatisch über VirtualDub rendern lassen

- Unter „Filme ins Pub“ werden Kopien aller Filme aus dem Ordner C:\Uselabordner\Filme in das öffentliche Verzeichnis des Usability-Labors gestellt. Eine entsprechende Mitteilung erscheint im Textfenster (siehe Abbildung B3 – 4).

Hinweis: Momentan wird nur die Mitteilung ausgegeben, hier passiert also nicht viel.

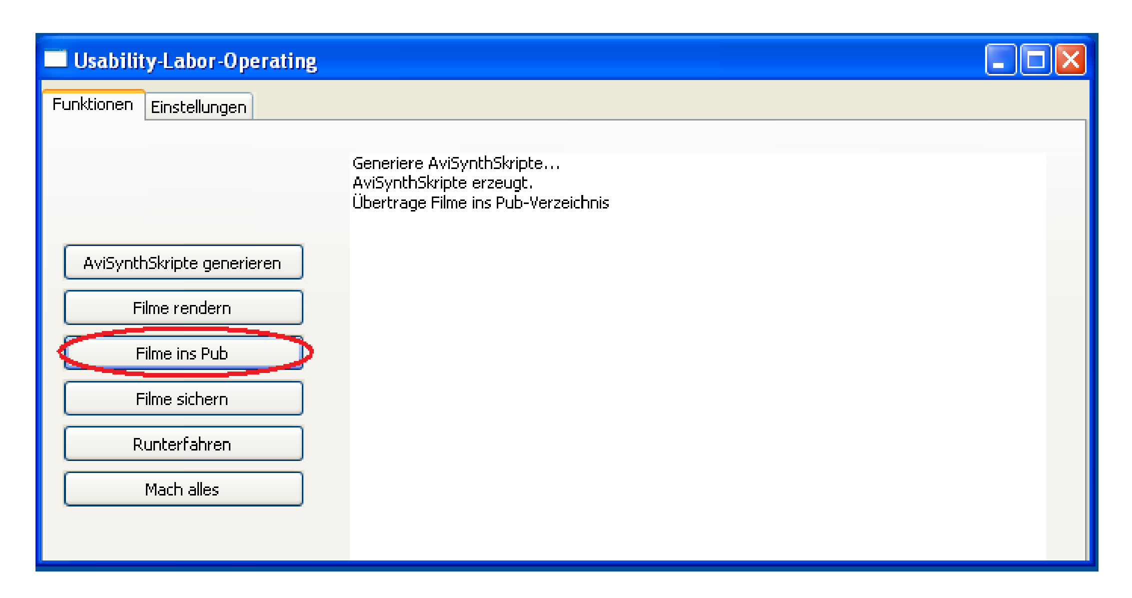


Abb. B3- 4: Filme in das öffentliche Verzeichnis transferieren

- „Filme sichern“: Entspricht dem Arbeitsschritt für die Archivierung von fertigen Filmen. Die Filme werden auf der Linkstation gesichert. Als Meldungen werden sowohl Pfad der Originaldatei sowie Pfad zur Linkstation für jeden Film ausgegeben (siehe Abbildung B3 – 5).

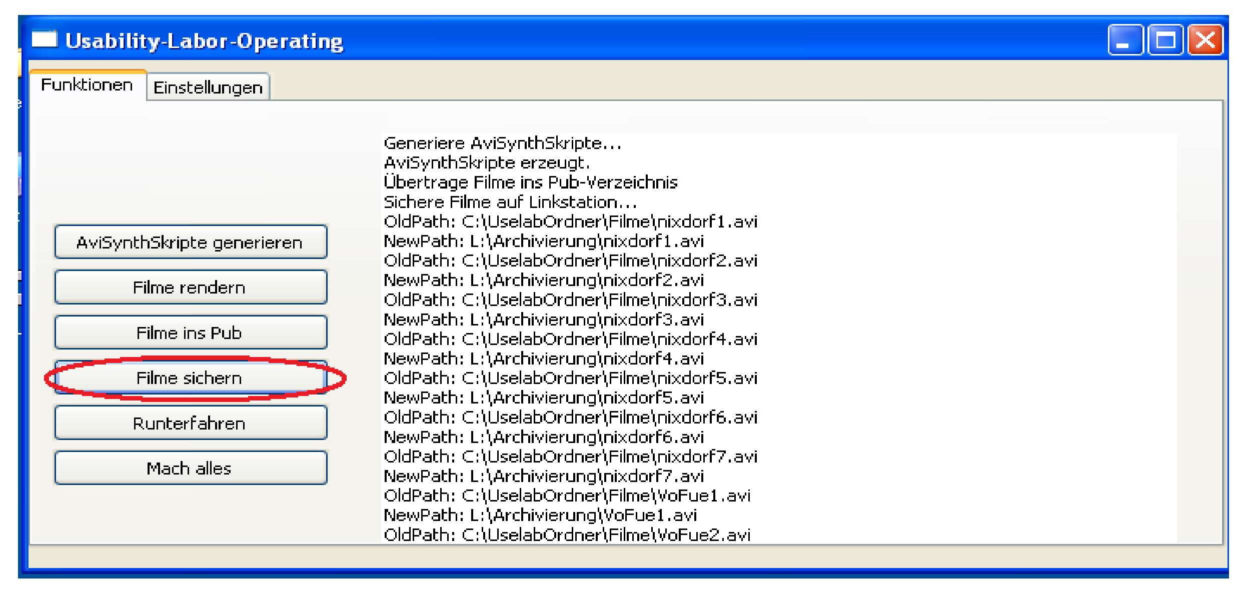


Abb. B3- 5: Filme auf der Linkstation sichern

- „Rechner runterfahren“: Fährt den Schnittrechner nach einer Wartezeit von 30 Sekunden runter (siehe Abbildung B3 – 6).
  Achtung: Es sollten alle noch laufenden Programme geschlossen werden oder die Beendigung der Programme abgewartet werden !!
  Alle nicht gespeicherten Daten gehen sonst verloren und noch laufende Anwendungen werden automatisch abgebrochen.
  Dies ist daher unbedingt nachzuprüfen.

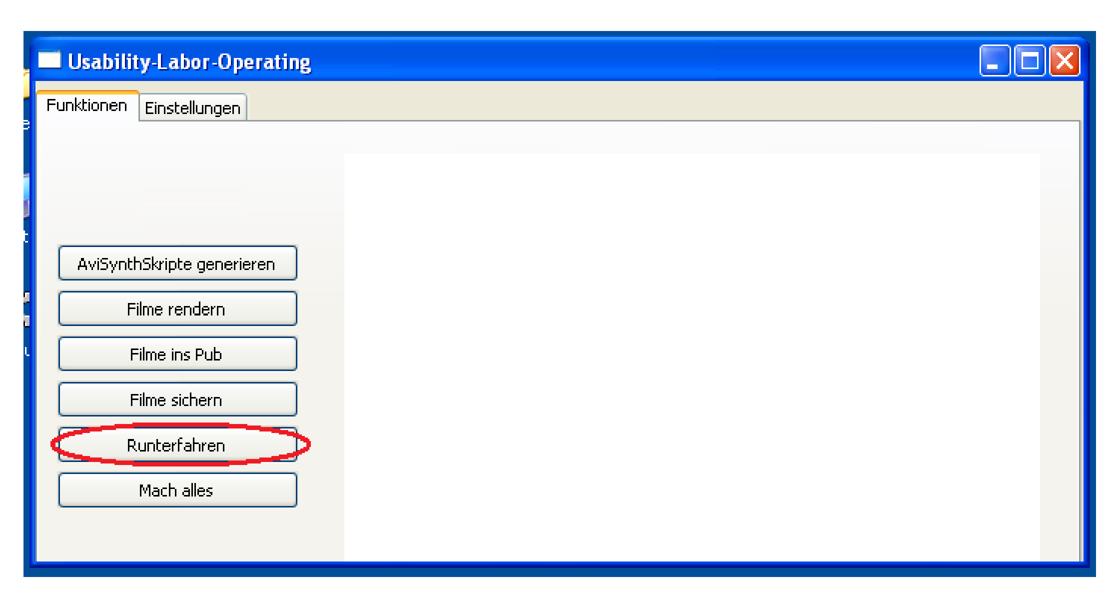


Abb. B3- 6: Rechner runterfahren

- „Mach alles“ führt nacheinander die einzelnen Schritte von der Generierung der Skripte bis einschliesslich zum Runterfahren des Rechners automatisch durch (siehe Abbildung B3 – 7).

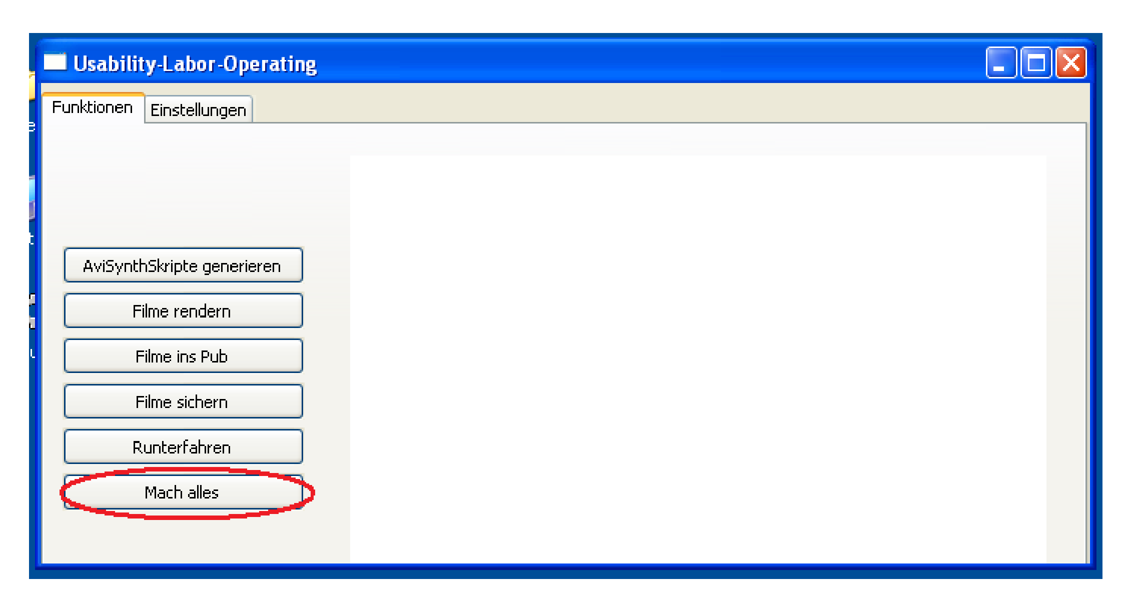


Abb. B3- 7: Komplettdurchlauf

B4 Acronis

1. Acronis True Image Workstation starten

Wahlweise wird Acronis über das Startmenü oder über die Desktop-Verknüpfung gestartet, siehe Abbildung B4 – 1.

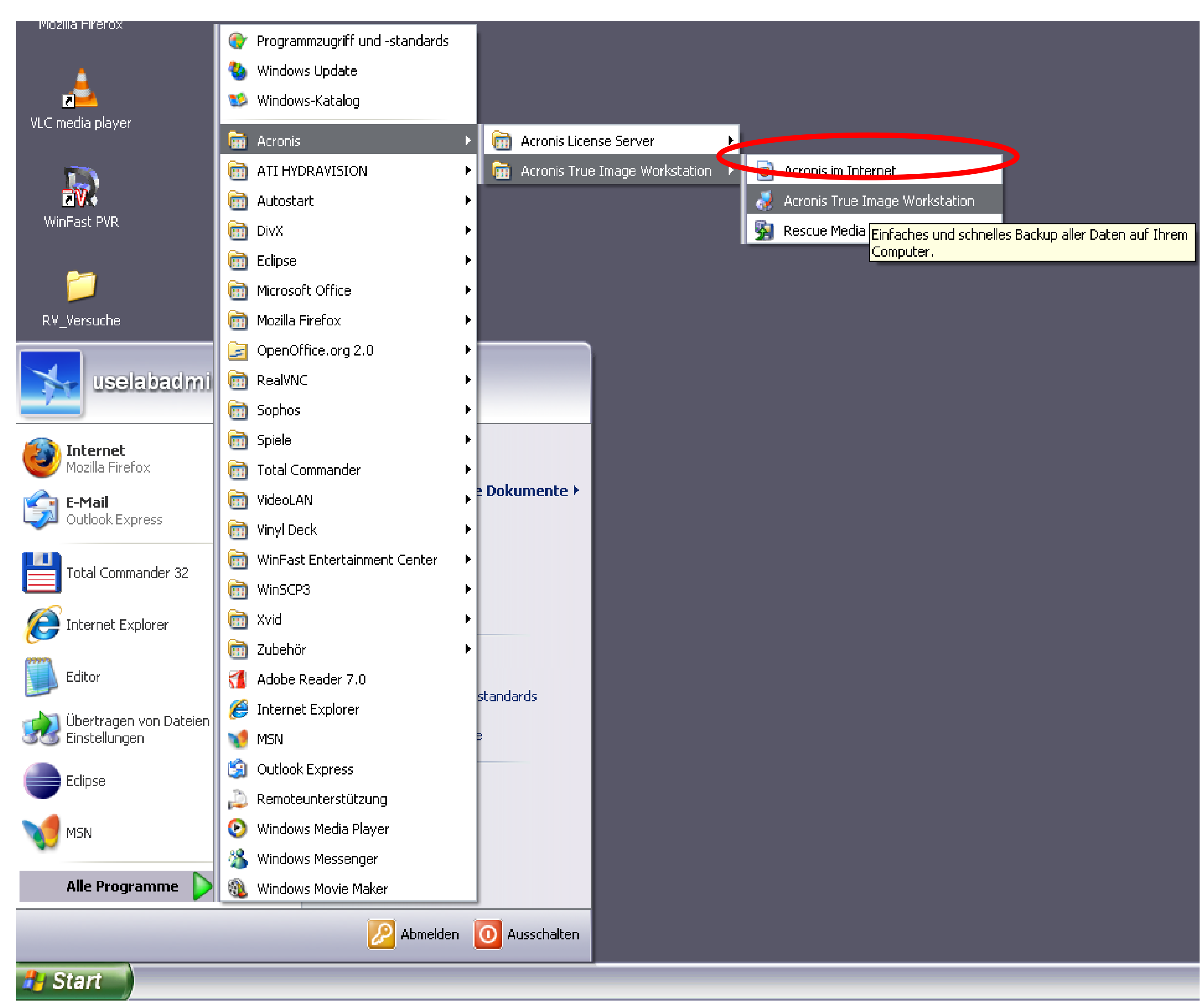


Abb. B4- 1: Acronis über Startleiste

2. Daten sichern

1. Backup auswählen (Abbildung B4 – 2)

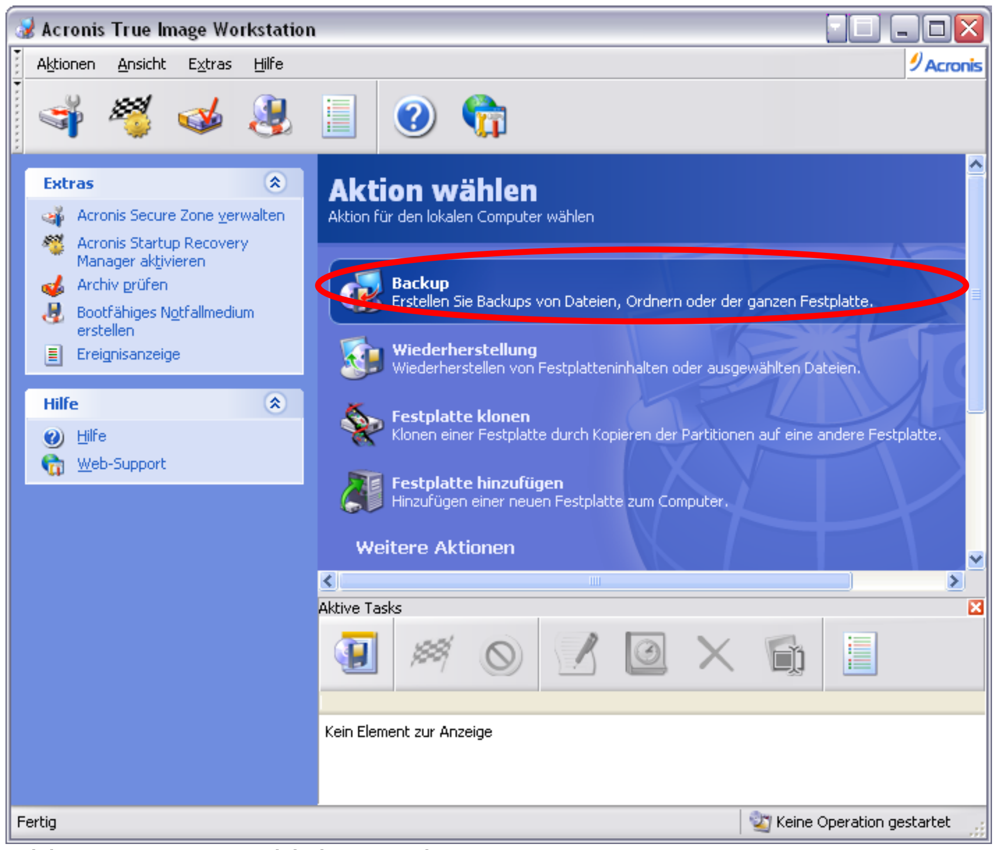


Abb. B4- 2: Auswahl des Backups

2. Den nächsten Dialogschritt mit „weiter“ bestätigen (Abbildung B4 – 3)

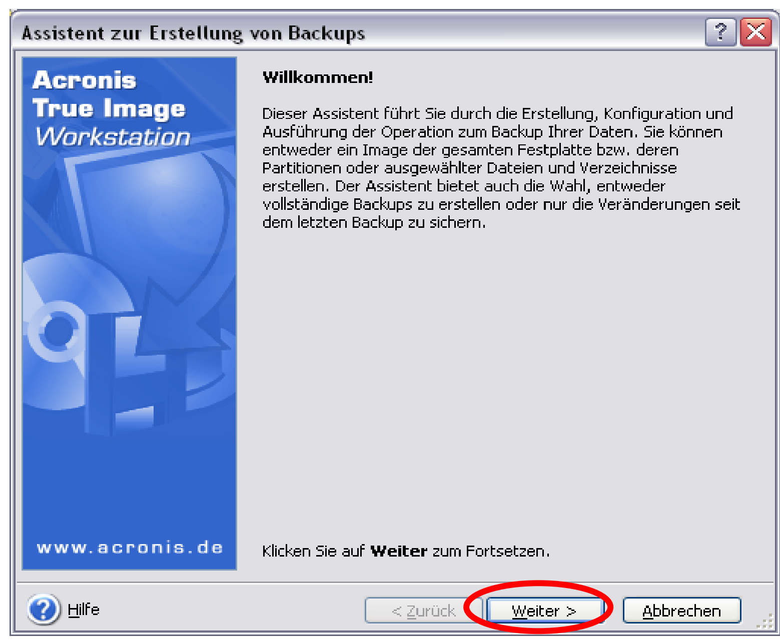


Abb. B4- 3: Begrüssungsbildschirm des Assistenten

3. Typ des Backups auswählen (Abbildung B4 – 4)

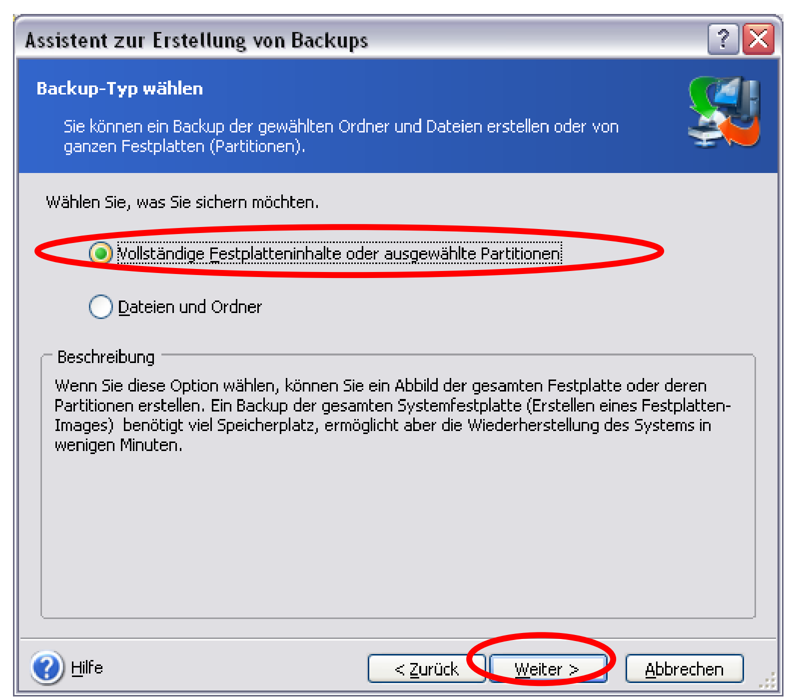


Abb. B4- 4: Art des Backups

- Backup des vollständigen Festplatteninhaltes bei Durchführung einer Komplettsicherung aller Festplatten
- Backup von Dateien und Ordnern für die Zweitsicherung der Endfilme

Anschliessend mit „weiter“ fortfahren.

4. Auswahl der Partition (Abbildung B4 – 5)

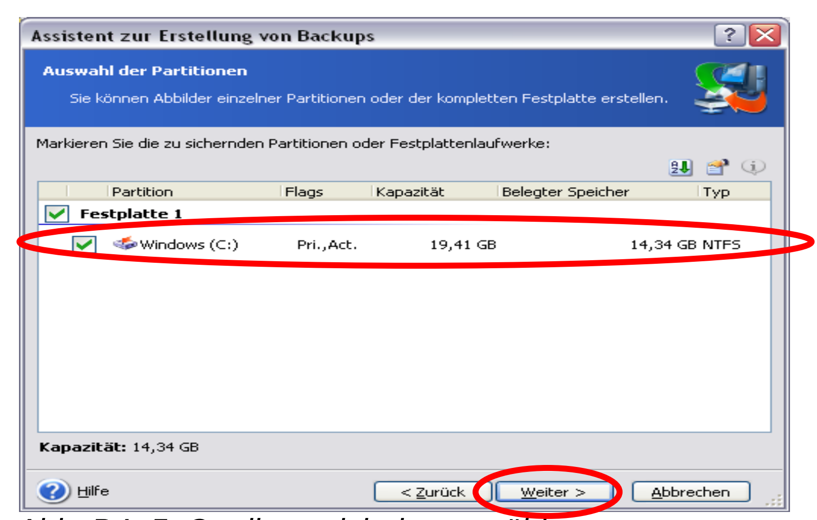


Abb. B4- 5: Quellverzeichnis auswählen

- Auf dem Testrechner sollte stets die C: - Partition gesichert werden
- Auf dem Schnittrechner sind die Partitionen C: und D: zu sichern
- Ebenfalls sollte das G: - Laufwerk der externen Festplatte gesichert werden
- Für den Filmeordner sollte das Verzeichnis L:\Archivierung ausgewählt werden (zur Erinnerung: dieses Verzeichnis befindet sich auf der Linkstation)

5. Zielverzeichnis auswählen (Abbildung B4 – 6)

Alle Backups sollten auf der Linkstation im Ordner L:\Backups gespeichert werden. Eine Ausnahme stellt das Backup des Filmeordners dar, da das Ursprungsverzeichnis selbst auf der Linkstation angegeben ist. Hier bietet sich beispielsweise die externe Festplatte G: zur Aufnahme der Backupdatei an.

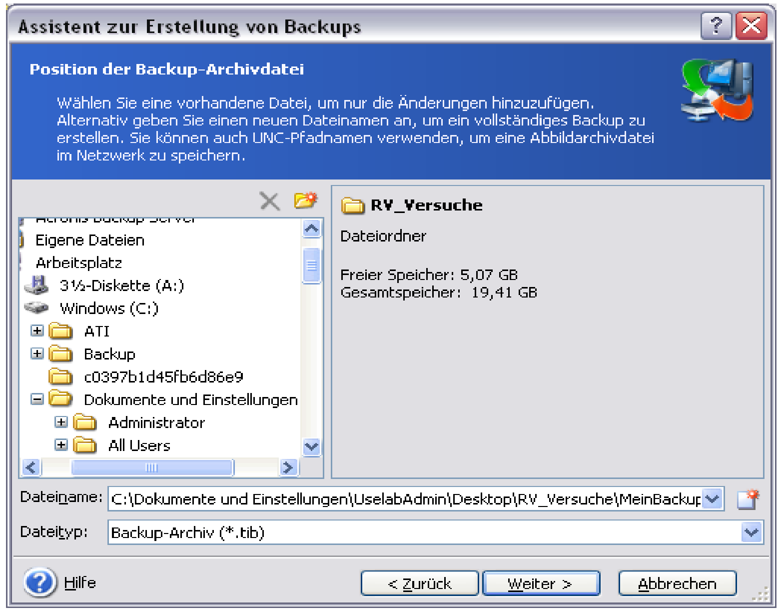


Abb. B4- 6: Zielpfad angeben

6. Backup-Modus wählen:

   Die Option „Erstelle ein neues vollständiges Backup-Archiv“ erstellt ein exaktes Abbild der gesamten Festplatte. Diese Option ist bevorzugt auszuwählen (vergleiche Abbildung B4 – 7).

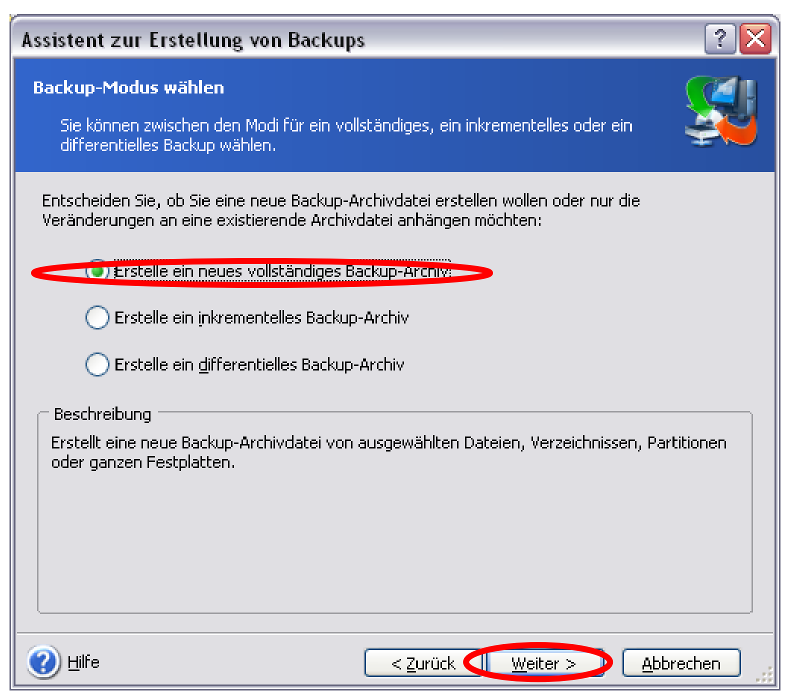


Abb. B4- 7: Auswahl des Archivtyps

7. Backup-Optionen wählen

   Standardoptionen verwenden (Abbildung B4 – 8). Bei der Installation wurden diese manuell so abgeändert, dass grundsätzlich die maximal mögliche Kompression auf die Daten angewandt wird.

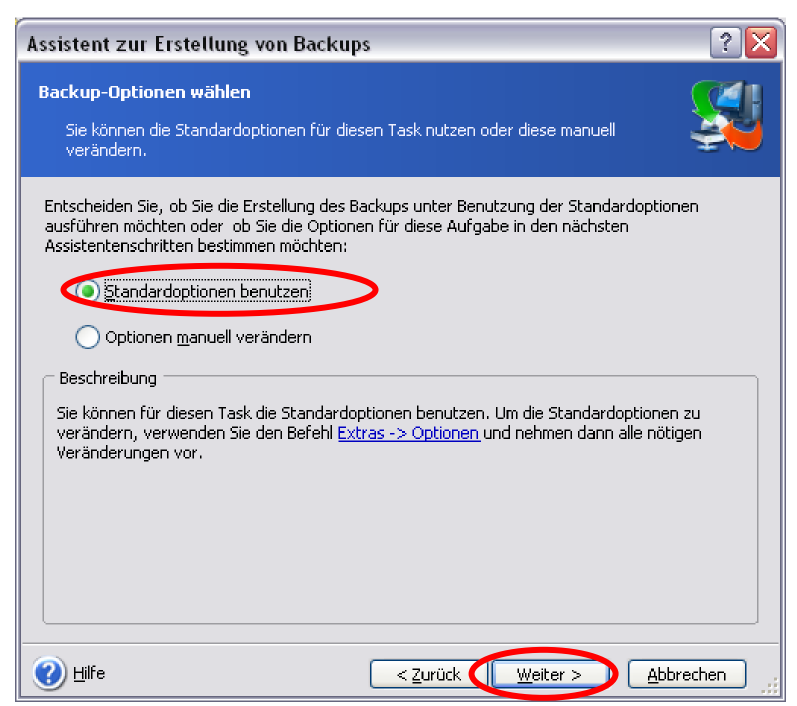


Abb. B4- 8: Auswahl der Optionen

8. Archiv Kommentieren (Abbildung B4 – 9)

   Hier sollte eine aussagekräftige Beschreibung des Backups angegeben werden.

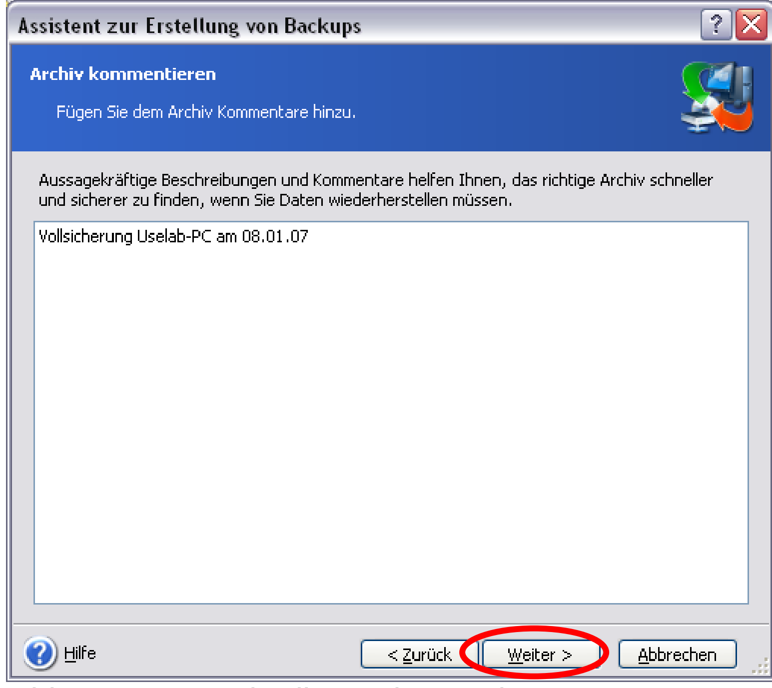


Abb. B4- 9: Beschreibung des Backups

9. Fertigstellen

   Zur Sicherheit nochmal die angegebenen Informationen nachprüfen. Dann „Fertigstellen“ auswählen und den Backup-Vorgang starten (Abbildung B4 – 10). Dies kann jetzt einige Zeit dauern.

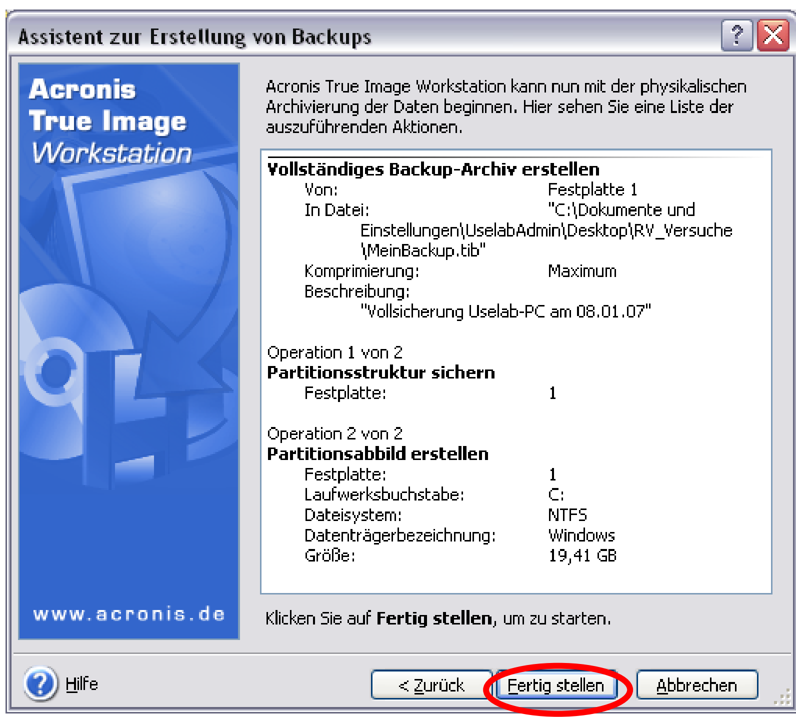


Abb. B4- 10: Starten des Backupvorgangs

3. Daten wiederherstellen

1. Bei Angabe der Aktionen „Wiederherstellung“ auswählen, siehe Abbildung B4 – 11.

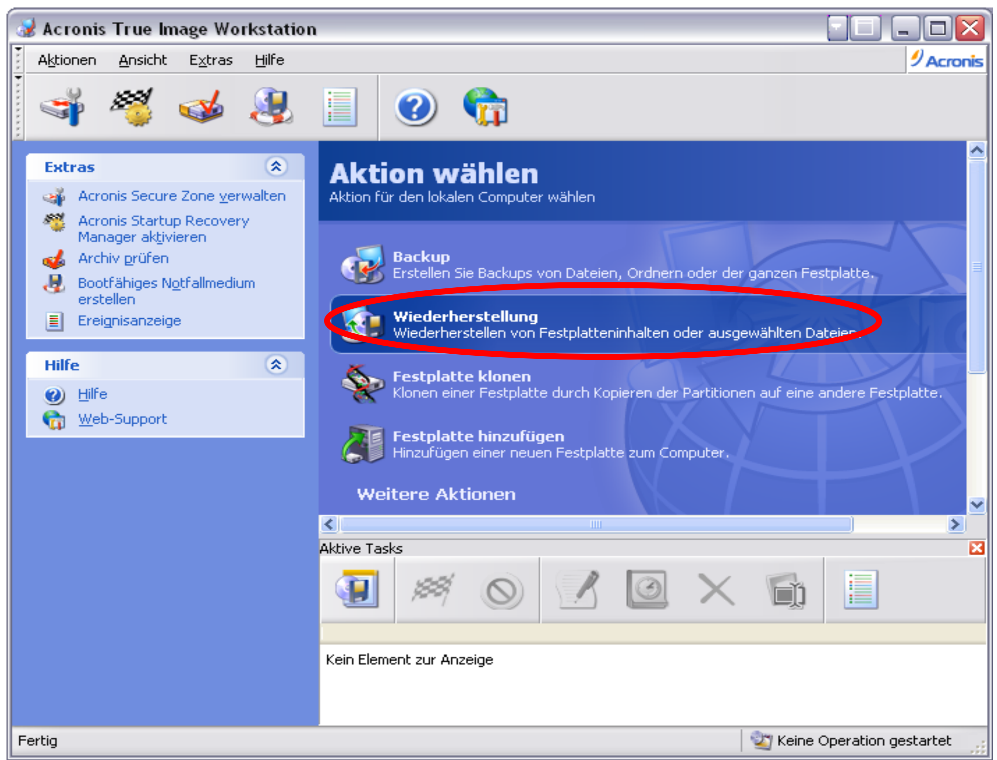


Abb. B4- 11: Festplatte soll wiederhergestellt werden

2. Der Assistent zur Wiederherstellung meldet sich. Im weiteren Verlauf den Anweisungen dieses Assistenten folgen.

B5 Mousetrace

Der Mousetracer kann über die ausführbare JAR – Datei mousetrace.jar bequem vom Desktop des Schnittrechners aus gestartet werden. Nach dem Start erscheint eine Benutzeroberfläche, wie sie in Abbildung B5 – 1 zu sehen ist.

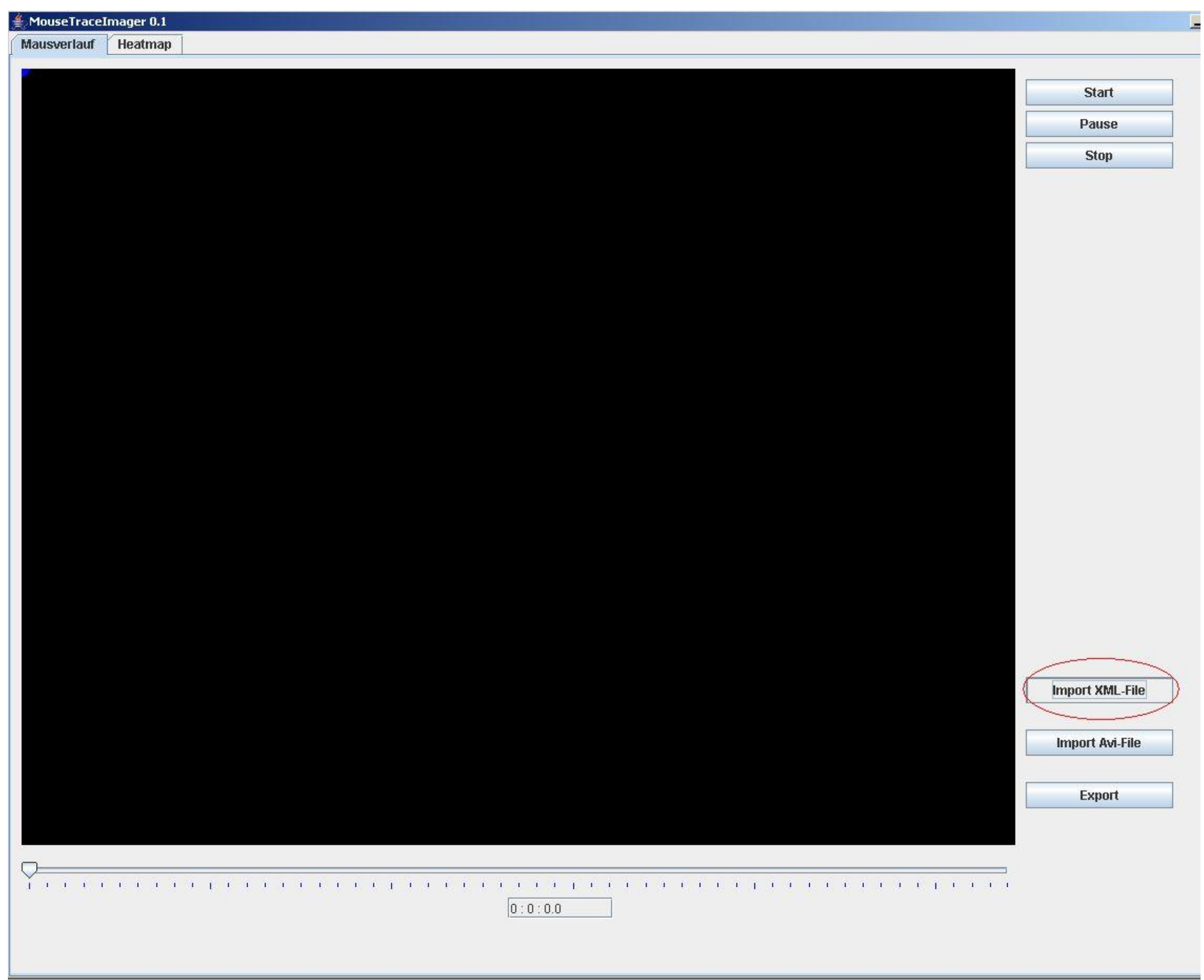


Abb. B5- 1: Mousetracer nach dem Starten

Nach dem Start hat man nun verschiedene Möglichkeiten zur Einstellung des Mousetracers. Zunächst einmal lässt sich über den Button „Import XML – File“ eine XML – Datei des Mousetrackers einlesen (siehe rote Markierung in der obigen Abbildung). Dies ist zwingend notwendig, wenn ein neuer Film über einen Mausverlauf dargestellt werden soll. Nach dem Laden wird bereits das erste Bild des Mausverlaufs angezeigt.

Über den Button „Import Avi – File“ lässt sich zusätzlich ein Hintergrundvideo (z.B. die zu dem Test gehörende Aufzeichnung des Eyetrackers) laden, das parallel während der Anzeige des Mausverlaufs mit angezeigt wird (Abbildung B5 – 2).
Dies ist jedoch nicht notwendig, es genügt, die erforderliche XML – Datei einzulesen. Gleichzeitig bietet dieser Button die Möglichkeit, ein bereits vorhandenes Video des Maustracers zur Anzeige zu laden.

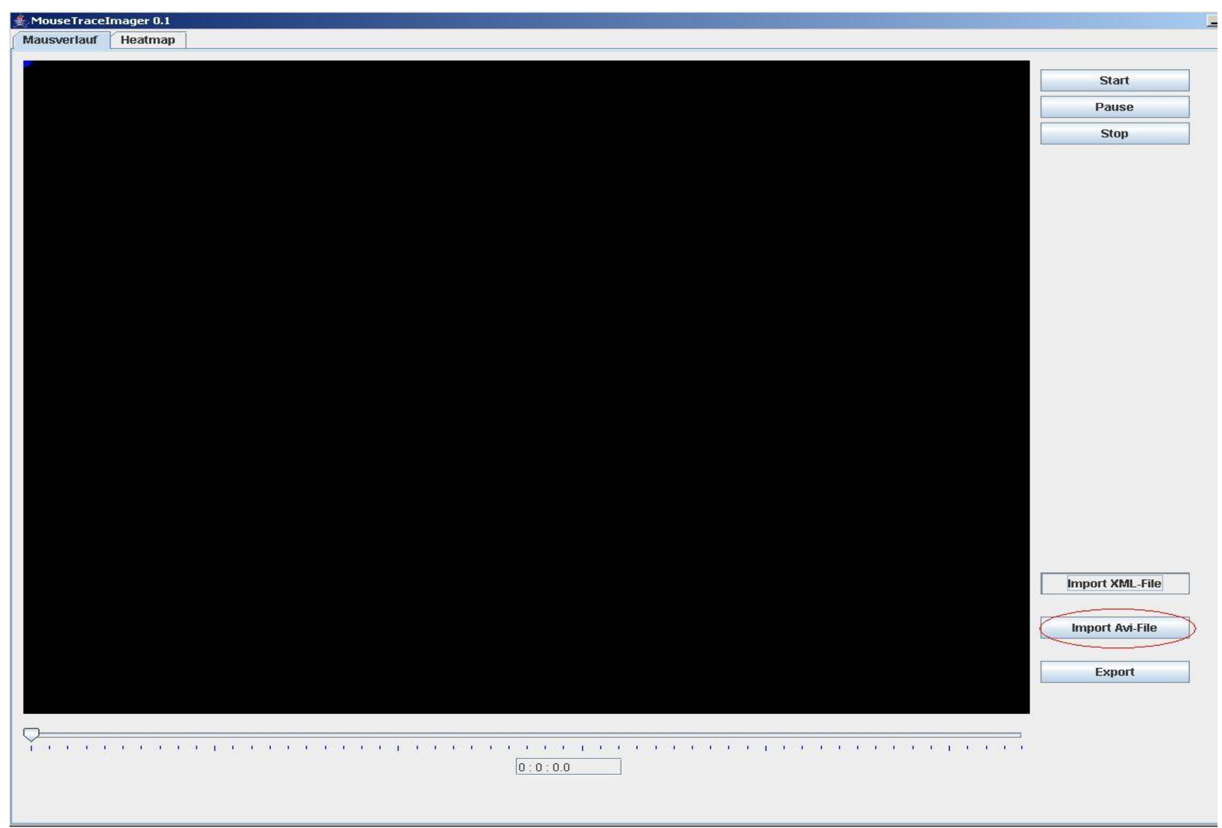


Abb. B5- 2: Importieren eines Hintergrundvideos

Über den „Export – Button“ können die Einzelbilder des Mausverlaufs als zusammengeschnittene AVI – Datei exportiert werden (Abbildung B5 – 3).

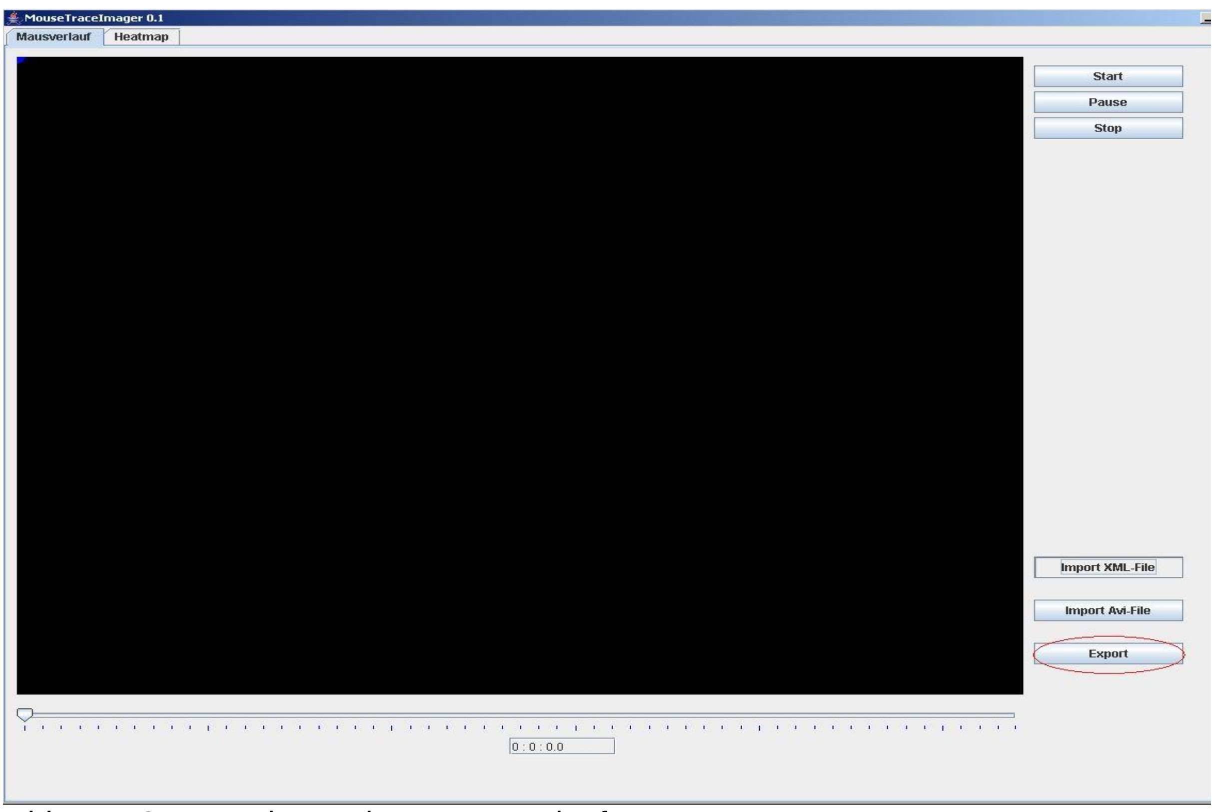


Abb. B5- 3: Exportieren eines Mausverlaufs

Der Fortschritt über den Mausverlauf lässt jederzeit über die Fortschrittsanzeige unterhalb des Bildbereichs anzeigen. Die verstrichene Zeit wird dabei in der kleinen Textbox festgehalten (siehe Abbildung B5 – 4).

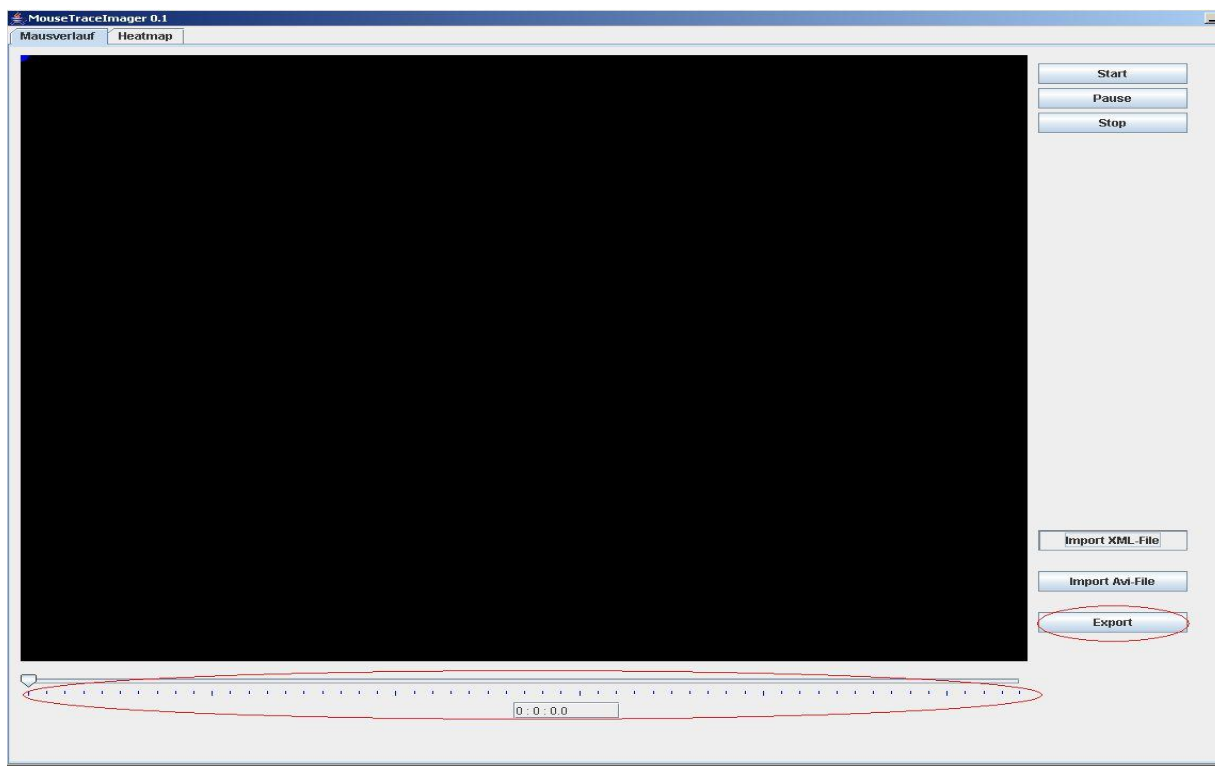


Abb. B5- 4: Fortschrittsanzeige

Zum Starten der Mausanzeige muss nun als nächstes der „Start“-Button betätigt werden (Abbildung B5 – 5).

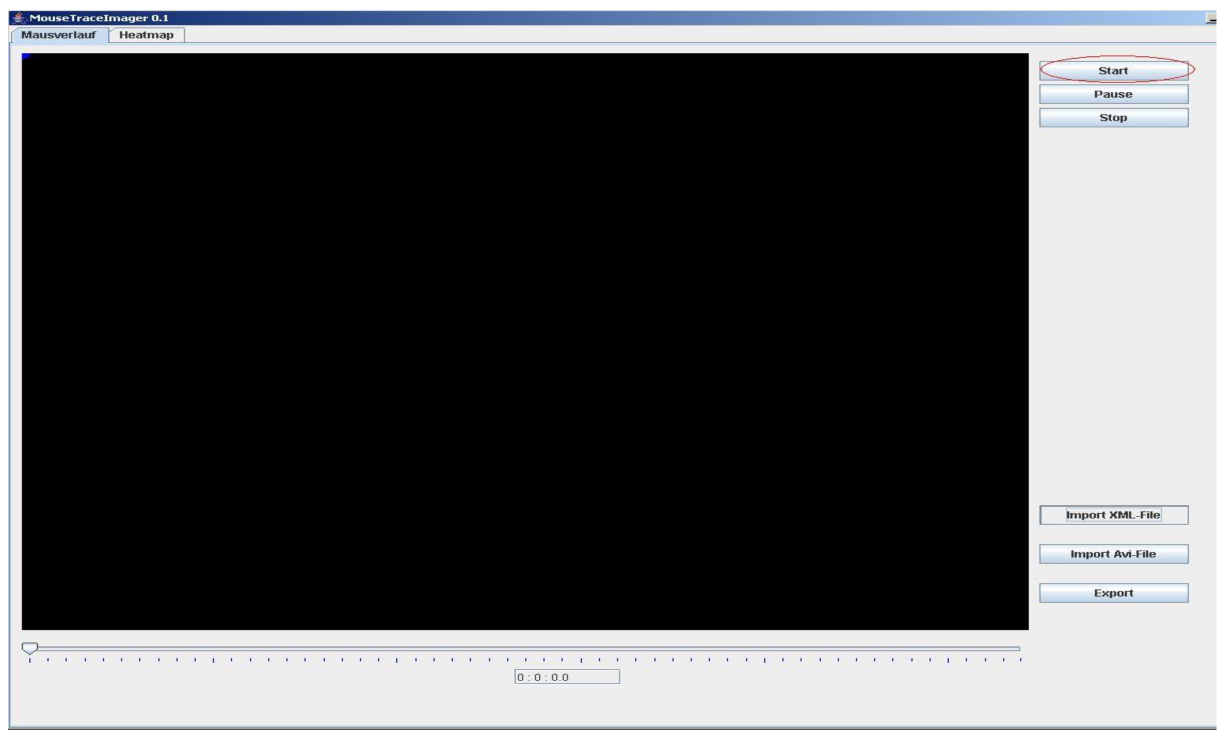


Abb. B5- 5: Start-Funktion in der Videoansicht

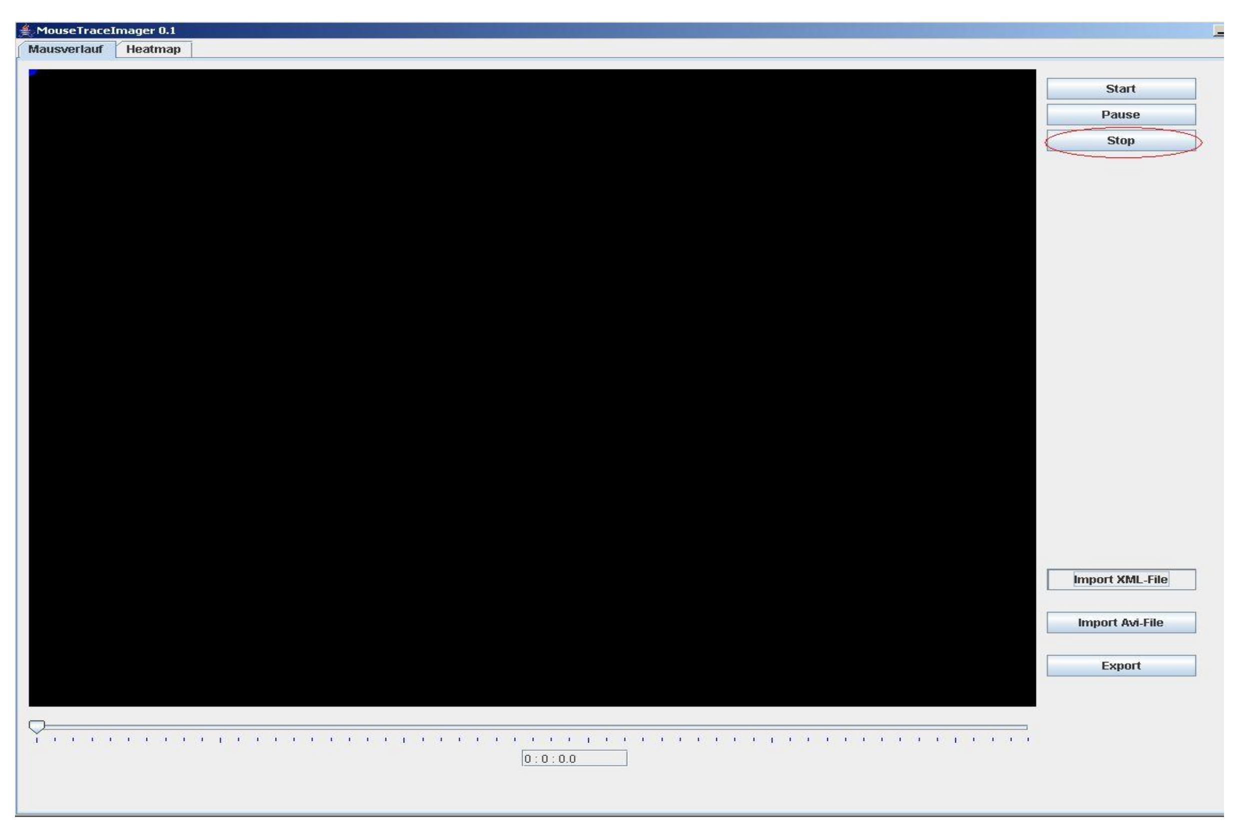


Abb. B5- 6: Stop-Funktion in der Videoansicht

Das Video über den Mausverlauf kann zu jeder Zeit über den „Stop“ – Button (Abbildung B5 – 6) oder über den „Pause“ – Button (Abbildung B5 – 7) angehalten werden.

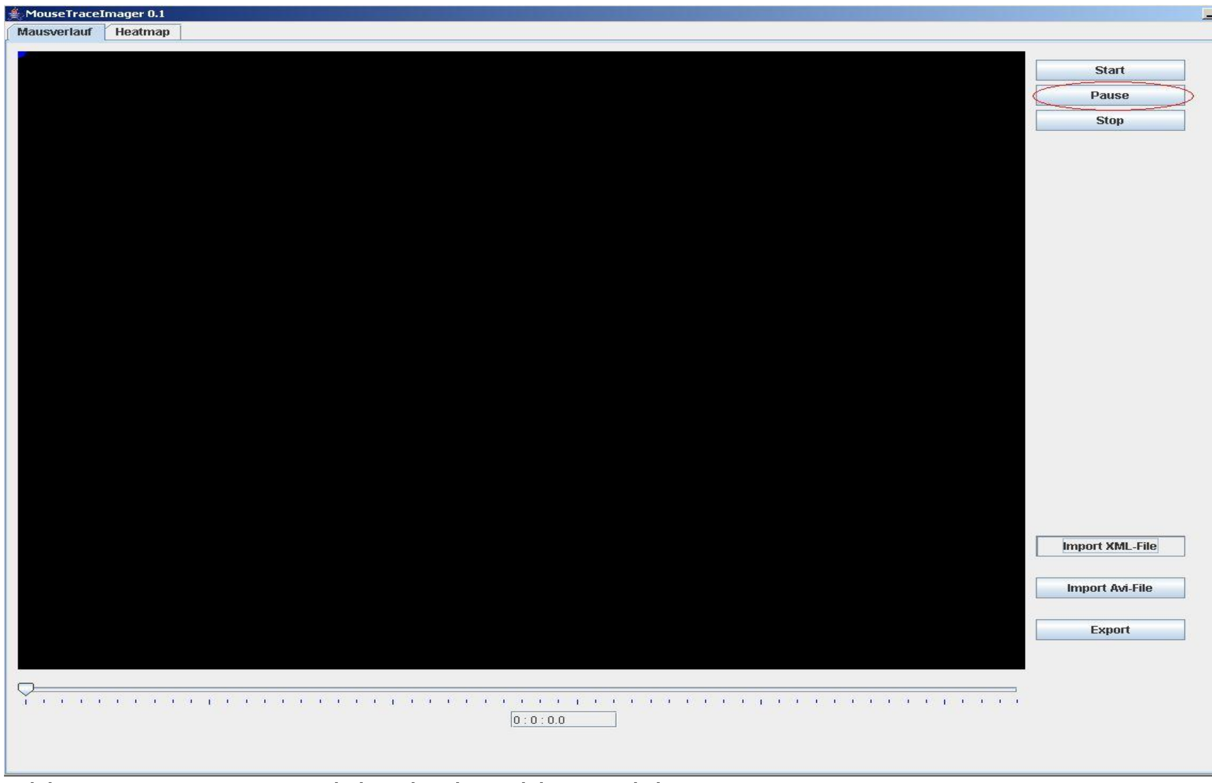


Abb. B5- 7: Pause-Funktion in der Videoansicht

Wird die Anzeige mittels „Pause“ angehalten, so kann die Aufzeichnung mit einem erneuten Klick auf „Pause“ von der letzten Stelle aus fortgesetzt werden. Wird die Anzeige dagegen mittels „Stop“ angehalten, so muss das Video neu gestartet werden.

Heatmap – Ansicht

Über den Tab „Heatmap“ gelangt man in die Heatmap – Ansicht. Hier hat man die Möglichkeit, Ballungsgebiete des Mausaufenthalts zu bestimmten Zeitpunkten zu ermitteln, die in Form von Wolken dargestellt werden. Über die beiden Slider lässt sich dabei das betrachtete Zeitfenster einstellen. Mit dem oberen Slider wird der früheste Zeitpunkt, mit dem unteren Slider der späteste Zeitpunkt eingestellt (Abbildung B5 – 8).

Achtung: Ist keine XML – Datei eingelesen worden, so kann auch keine Heatmap gezeichnet werden. In diesem Fall sind die Slider nicht verstellbar. Es ist ebenfalls nicht möglich, den oberen Slider auf eine Position hinter dem unteren Slider zu setzen (der früheste Zeitpunkt kann nicht hinter dem spätesten Zeitpunkt liegen).

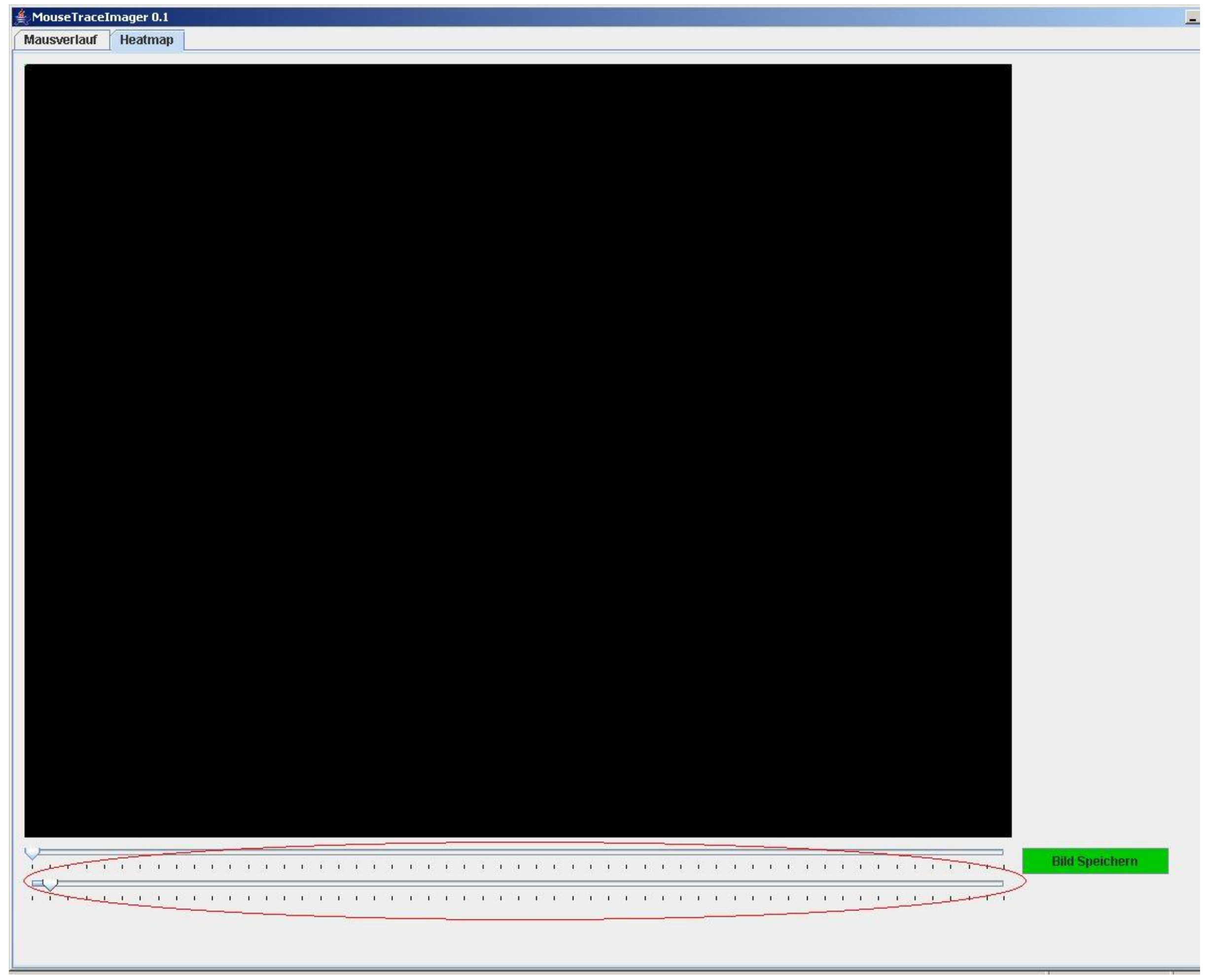


Abb. B5- 8: Slider in der Heatmap-Ansicht

Zusätzlich kann jedes einzeln angezeigte Bild als PNG – Datei abgespeichert werden. Dies ist über den Button „Bild speichern“ möglich (Abbildung B5 – 9).

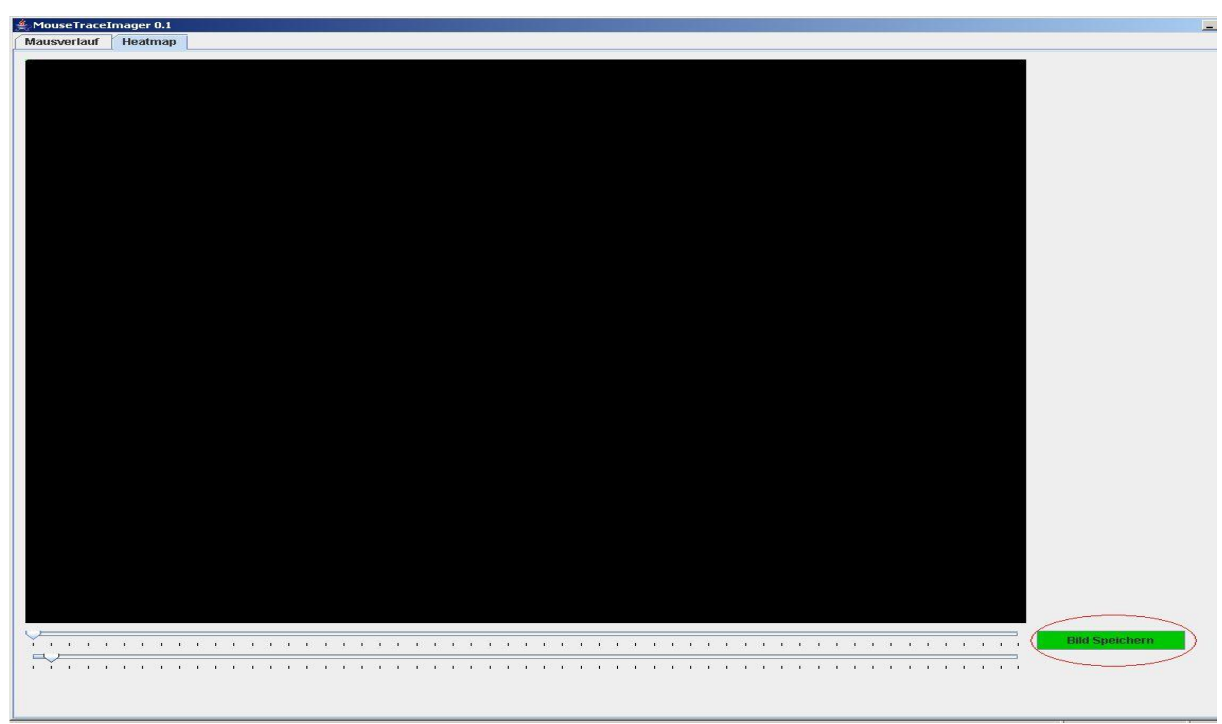


Abb. B5- 9: Speicherfunktion in der Heatmap-Ansicht

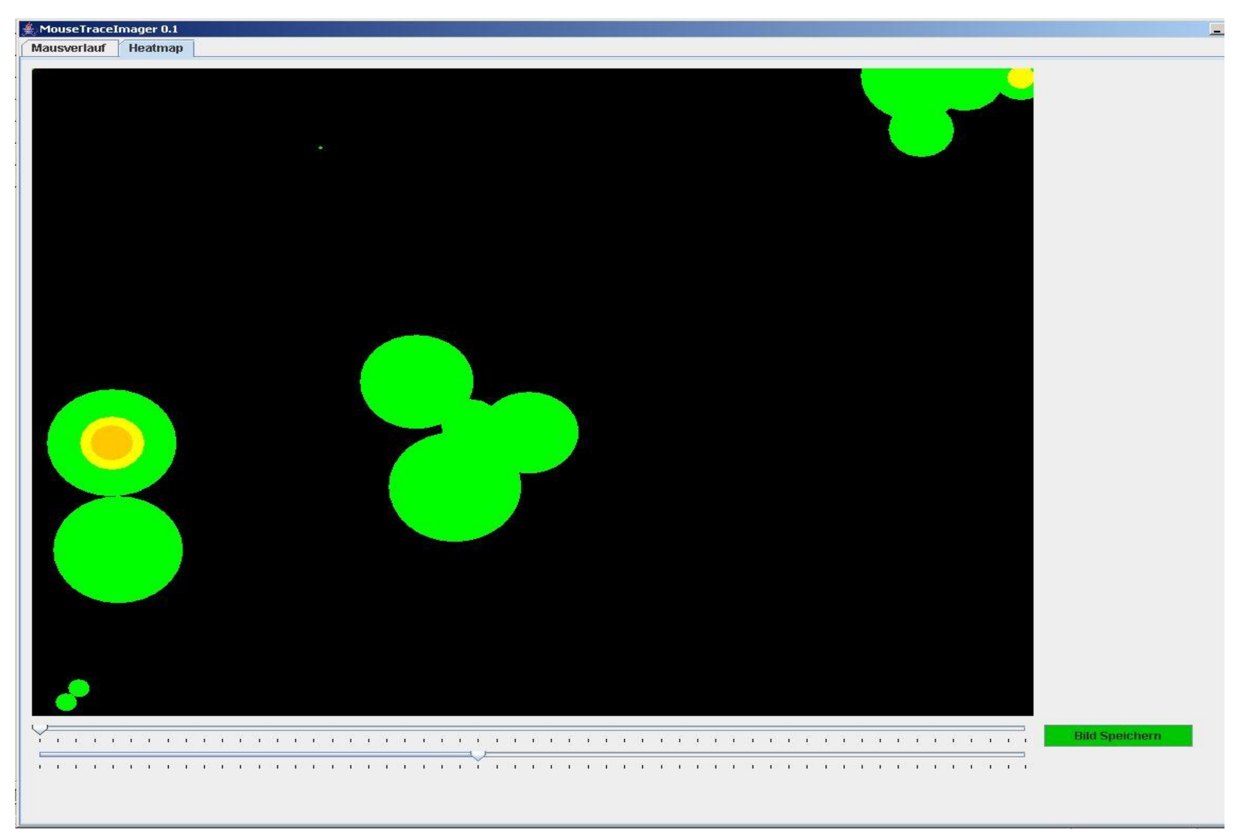


Abb. B5- 10: Heatmap im Mousetracer

Bei ordnungsgemässer Handhabung sollte eine Heatmap schliesslich etwa wie in Abbildung B5 – 10 dargestellt werden.

B6 Bedienung des Gesamtsystems

1. Das System hochfahren

Im Testraum:

- Testrechner mitsamt Soundsystem einschalten und hochfahren. Welcher der beiden Testrechner verwendet wird, hängt vom Untersuchungsgegenstand ab. Passwörter notfalls beim Professor oder zuständigen Mitarbeiter erfragen. Software von Buzzertool und Mousetracker werden automatisch gestartet. Das Mousetracker – Fenster rechts aus dem Bildschirmbereich ziehen.
- Eyetracker einschalten, falls Verwendung erwünscht ist. Gleichzeitig die Software des Eyetrackers starten (Tobii Studio) und einrichten. Nähere Einzelheiten hierzu sind der Diplomarbeit von Stefan Richter zu entnehmen.
- Kameras mit der Fernbedienung aus kürzester Entfernung einschalten. Anmerkung: Die Kameras 1 bis 5 reagieren auf den Kanal 1, die Kamera 6 hingegen auf den 2.Kanal.

Im Regieraum:

- Schnittrechner (im Turm querliegend) und Capturerechner (über dem Schnittrechner) in der Reihenfolge von unten nach oben einschalten und hochfahren. Passwörter notfalls beim Professor oder zuständigen Mitarbeiter erfragen. Monitore einschalten.
- Falls benötigt, Linkstation einschalten (ganz oben im Rechnerturm)
- Verbindung Schnittrechner → externe Festplatte G: herstellen durch Öffnen des G:\ – Verzeichnisses auf dem Schnittrechner.
- Gleichen Vorgang auf den Capturerechnern wiederholen. Dies ist insbesondere für die Videoaufzeichnung von Bedeutung, da im Falle einer nicht korrekten Verbindung keine Speicherung der Kamerastreams auf der externen Festplatte erfolgt !
  Hinweis: Über die linke oder rechte STRG – Taste kann zwischen den einzelnen Rechnern hin – und hergewechselt werden.
- Auf beiden Capturerechnern das Aufzeichnungsprogramm Focus Capture Suite öffnen
- Metriken/Video – Switch einschalten
- Audio System Pioneer einschalten und Lautstärke auf - 31 dB einstellen
- Ton einschalten durch Einstecken des Steckers an der Mikrofonsteuerung
- Mikrofonsteuerung: Regler auf Mittelstellung bringen
- VGA – Bildschirm sowie Metriken – Bildschirm einschalten
- VGA – Steuerung einschalten
- Alle 6 Kamera – Monitore einschalten
- Positionen der Kameras über Kamerasteuerung einstellen
  Hinweis: Grundposition ist auf Taste 1 gespeichert, Position der Nachbesprechung auf Taste 2 der Kamerasteuerung

Wenn alles hochgefahren ist, sollte im nächsten Schritt ein Soundcheck durchgeführt werden. Es ist zu prüfen, ob Mikrofone und Lautsprecher sowohl im Testraum als auch im Regieraum ihre Funktionalität erfüllen.

Nachbedingungen:

- Ton ist in beiden Räumen hörbar
- Die Kameramonitore zeigen das Bild der jeweiligen Kamera
- Die Kamerabilder im Focus Capture Suite auf den Capturerechner sind identisch mit den Bildern von den Kameramonitoren und frei von Übertragungsstörungen
- Die Kameras ändern ihre Position entsprechend den Steuerungen über die Kamerafernbedienung
- Auf dem VGA – Bildschirm im Regiebild ist das exakte Abbild des Bildschirms vom Testrechner zu sehen, ebenso im Focus Capture Suite.
  Hinweis zur Software Focus Capture Suite:
  Auf dem 1. Capturerechner links werden die Kameras 1 – 4 angezeigt, in der Reihenfolge von rechts nach links und oben nach unten (also Kamera 1 oben links, Kamera 4 unten rechts). Auf dem 2. Capturerechner rechts werden auf der rechten Seite die Kameras 5 (oben) und 6 (unten) angezeigt, das VGA – Bild sollte auf der unteren linken Seite zu sehen sein. Der Quadrant oben links wird nicht verwendet. Anhand der Buttons „Sync“ kann eingestellt werden, welche Kamerafilme aufgenommen werden sollen. „Inlay“ schaltet die Anzeige des jeweiligen Kamerabildes ein bzw. aus.
- Auf dem Metriken – Monitor ist die Anzeige des Mousecapturers zu sehen

2. Aufzeichnung eines Usability – Tests

Vor der Aufzeichnung sind folgende Programme auf dem Schnittrechner zu starten:

- Client des Mousecapturers
  Eine entsprechende Verknüpfung liegt auf dem Desktop bereit.
  Anschliessend den Client mit dem Server (Testrechner) über das Menü Verbindung verbinden.
  Eine ausführliche Bedienungsanleitung des Mousecapturers kann der zugehörigen Dokumentation entnommen werden

- Client des Buzzertools
  Entsprechende Verknüpfung liegt auf dem Desktop bereit.
  Anschliessend Einstellungen wie in der entsprechenden Anleitung vornehmen.

- Zum ferngesteuerten Ansprechen der Aufnahmesoftware von Focus auf den Capturerechnern die Batch-Datei RunAll (liegt auf dem Desktop) starten. Daraufhin werden zwei Benutzeroberflächen angezeigt, mit denen die Aufzeichnung, die von den Capturerechnern durchgeführt wird, gestartet oder gestoppt wird.
  Achtung: Zuvor muss auf den beiden Capturerechnern die angesprochene Software geöffnet sein, da sie sonst nicht vom Schnittrechner aus angesprochen werden kann !!!

Anschliessend folgende Einstellungen vornehmen:

1. In den Feldern „Filename“ jeweils einen Dateinamen angeben
2. Das Feld „Channel“ beider Oberflächen muss auf 0 eingestellt sein
3. Im Adressfeld jeweils IP – Adressen der Capturerechner eingeben, für den linken Capturerechner die 10.0.2.19, für den rechten die 10.0.2.21.

Siehe auch Abbildung B6 – 1.

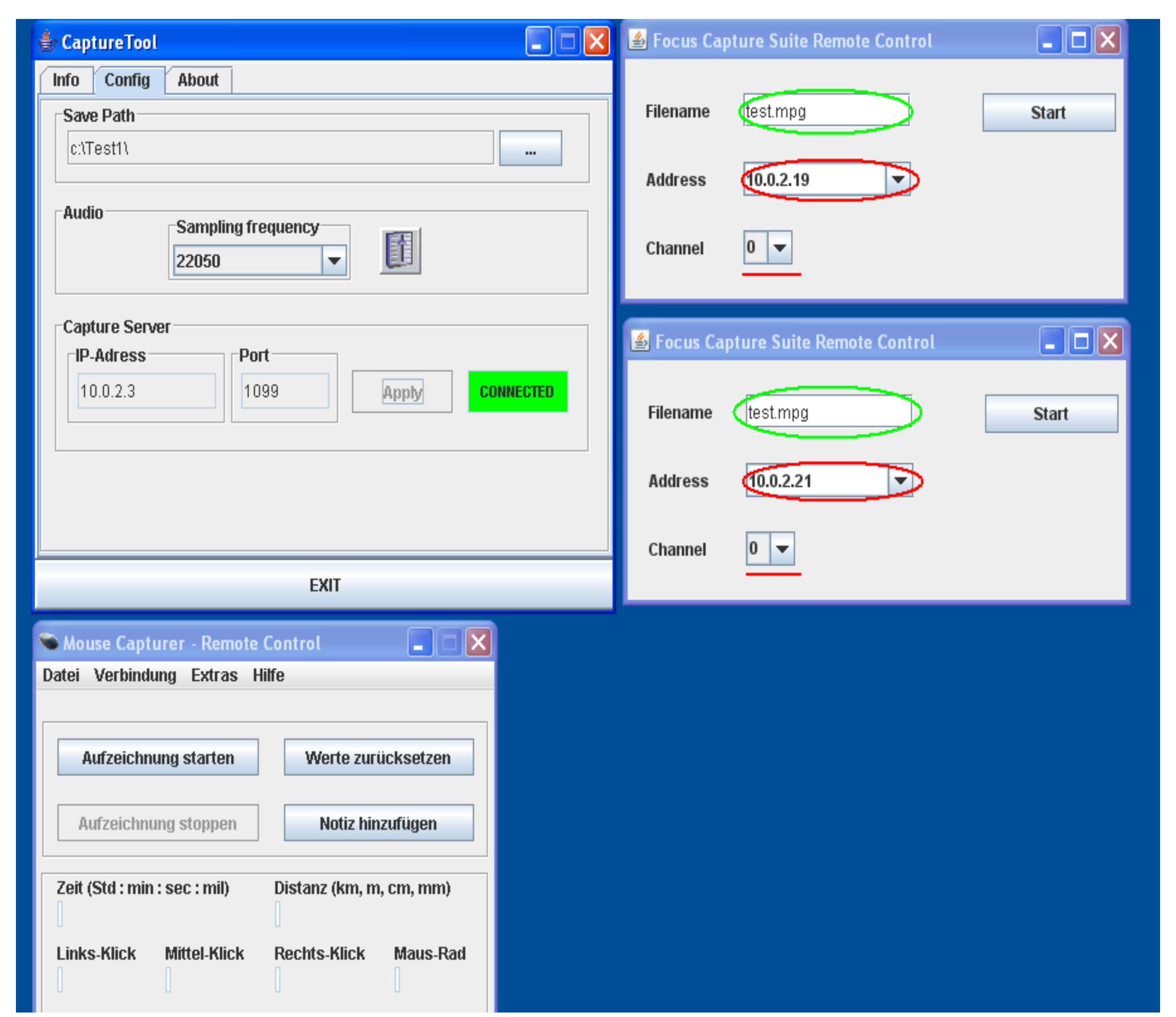


Abb. B6- 1: Voreinstellungen für die Videoaufzeichnung

Der Desktop des Schnittrechners sollte nun ungefähr der Abbildung B6 – 2 entsprechen.

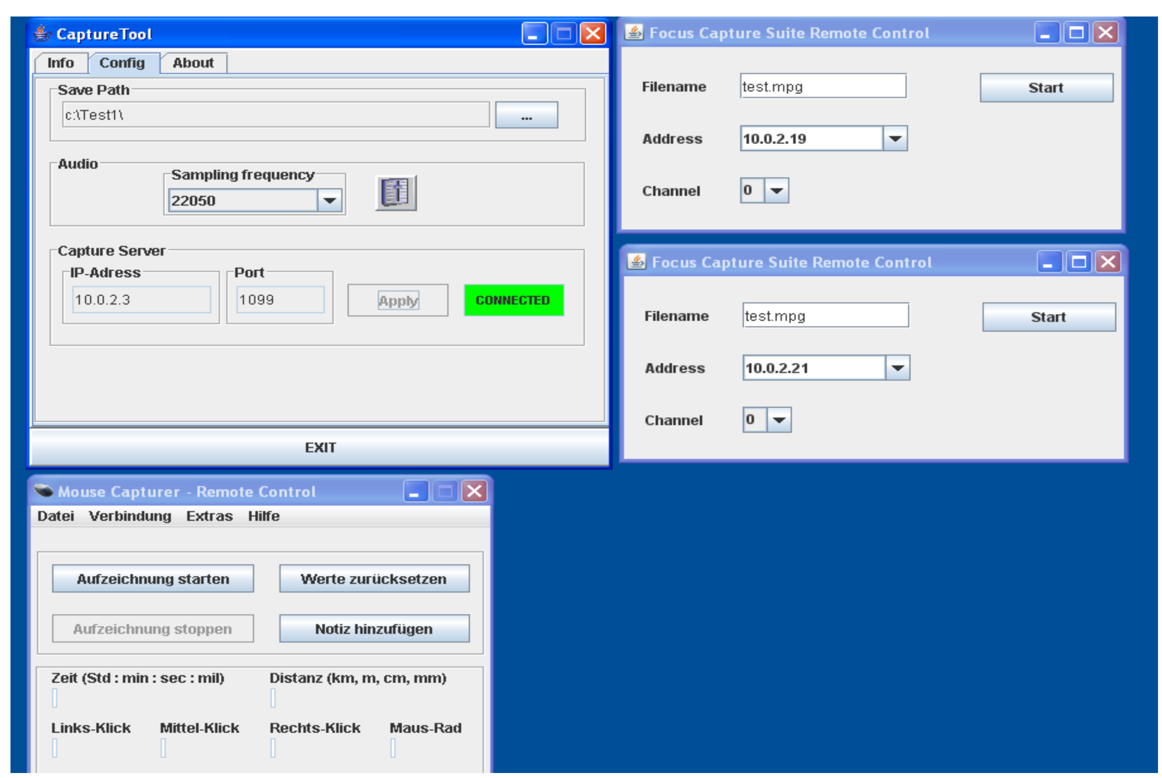


Abb. B6- 2: Benötigte Programme für die Aufzeichnung

Um die Aufzeichnung zu starten, sind nun die jeweiligen Startbuttons sowohl der beiden Fernsteuerungstools als auch des MouseCapturers innerhalb kürzester Zeit zu drücken (siehe Abbildung B6 – 3). Die Startbuttons der Fernsteuerungstools ändern daraufhin ihre Anzeige auf „Stop".

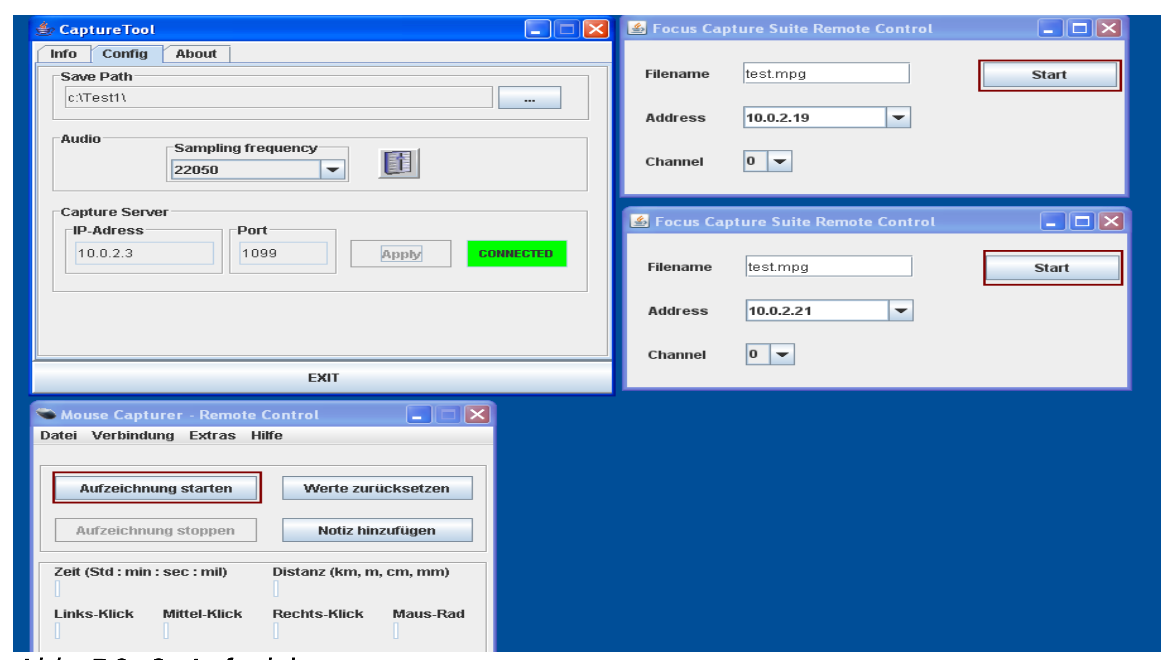


Abb. B6- 3: Aufzeichnung starten

Um die Aufzeichnung zu stoppen, sind die jeweiligen Stopbuttons sowohl der beiden Fernsteuerungstools als auch des MouseCapturers innerhalb kürzester Zeit zu drücken (siehe Abbildung B6 – 4). Die Startbuttons der Fernsteuerungstools ändern daraufhin ihre Anzeige zurück auf „Start".

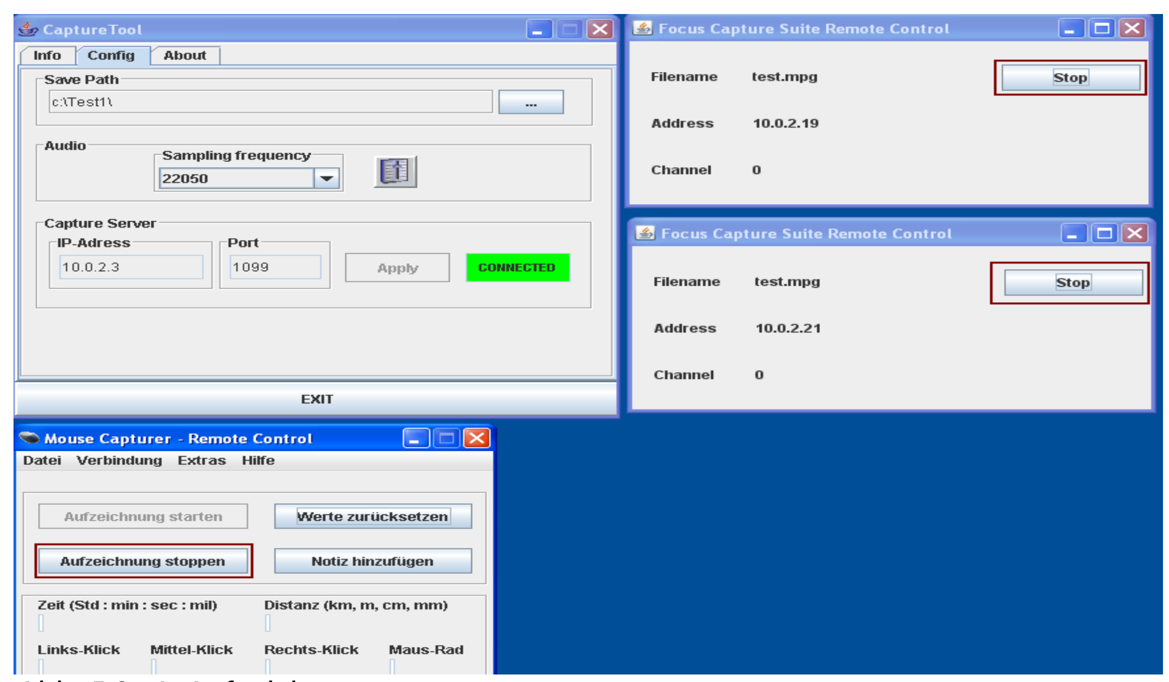


Abb. B6- 4: Aufzeichnung stoppen

Achtung !!
Es ist unbedingt erforderlich, direkt auf beiden Capturerechnern nachzuprüfen, ob die Aufzeichnung auch wirklich angehalten wurde. Dies ist nicht immer der Fall. Erkennbar ist dies an dem Aufleuchten des roten Kreises in der Capture Suite (siehe auch grüne Markierung in der Abbildung B6 – 5). In solchen Situationen unbedingt den Button direkt daneben im 1. Quadranten manuell betätigen (siehe rote Markierung).

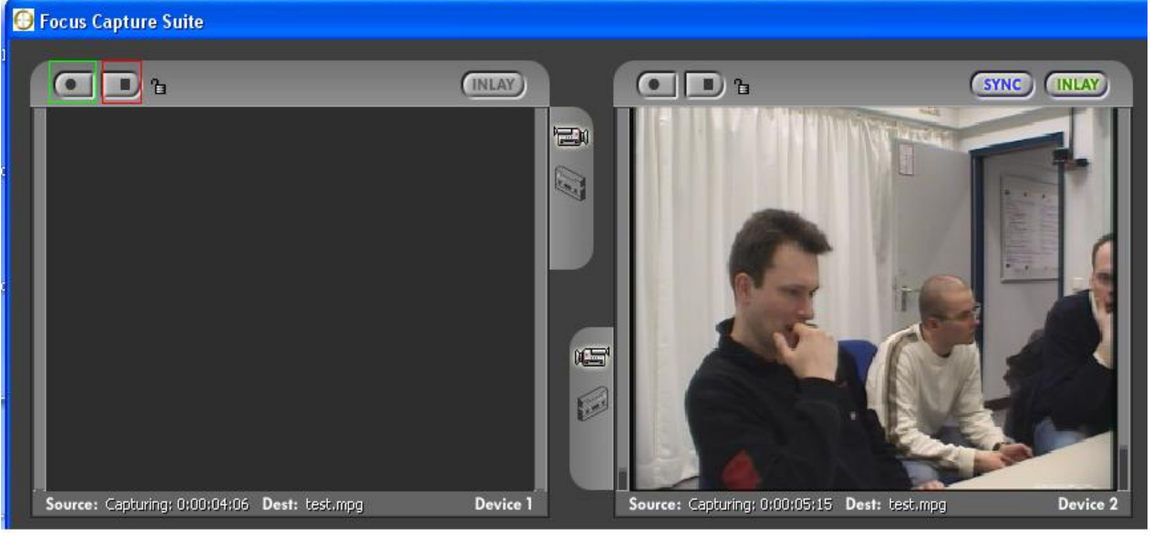


Abb. B6- 5: Focus Capture Suite auf dem Videograbber

Nach der Videoaufzeichnung sollten die Daten des Mousecapturers für die spätere Auswertung gesichert werden. Hierzu über das Menü „Datei“ gehen, den Punkt „Aufzeichnung exportieren (XML)“ wählen und die Datei in einem geeigneten Verzeichnis abspeichern (am besten in dem Ordner, der auch für die Daten des Buzzertools verwendet wird, dann hat man alles zusammen in einem Ordner). Anschliessend die Werte über den gleichnamigen Button auf 0 zurücksetzen.

3. Das System herunterfahren

Um das System herunterzufahren, müssen einfach die Schritte aus dem Punkt 1. In umgekehrter Reihenfolge durchgeführt werden. Dabei sind folgende Abweichungen zu berücksichtigen:

- Die Linkstation ist erst dann auszuschalten, wenn die Archivierung vollständig durchgeführt wurde
- Wenn direkt im Anschluss an die Aufzeichnung die Nachbearbeitung der Videos erfolgt (schneiden, rendern), den Schnittrechner erst nach vollständiger Videobearbeitung herunterfahren. Sollte das Operatingtool benutzt werden, wird der Rechner automatisch heruntergefahren.

B7 Any Video Converter

Für die Konvertierung der MPEG – Filme in ein anderes Format sind folgende Schritte auszuführen:

1. Über den Menüpunkt „Video hinzufügen“ jeden zu konvertierenden Film in den Converter laden. Informationen über den Film werden dann in der Liste angezeigt.
   Für jeden Film können zusätzlich Einstellungen wie Bildgrösse und Bildrate angepasst werden. Siehe auch entsprechende Markierungen in der folgenden Abbildung B7 – 1.

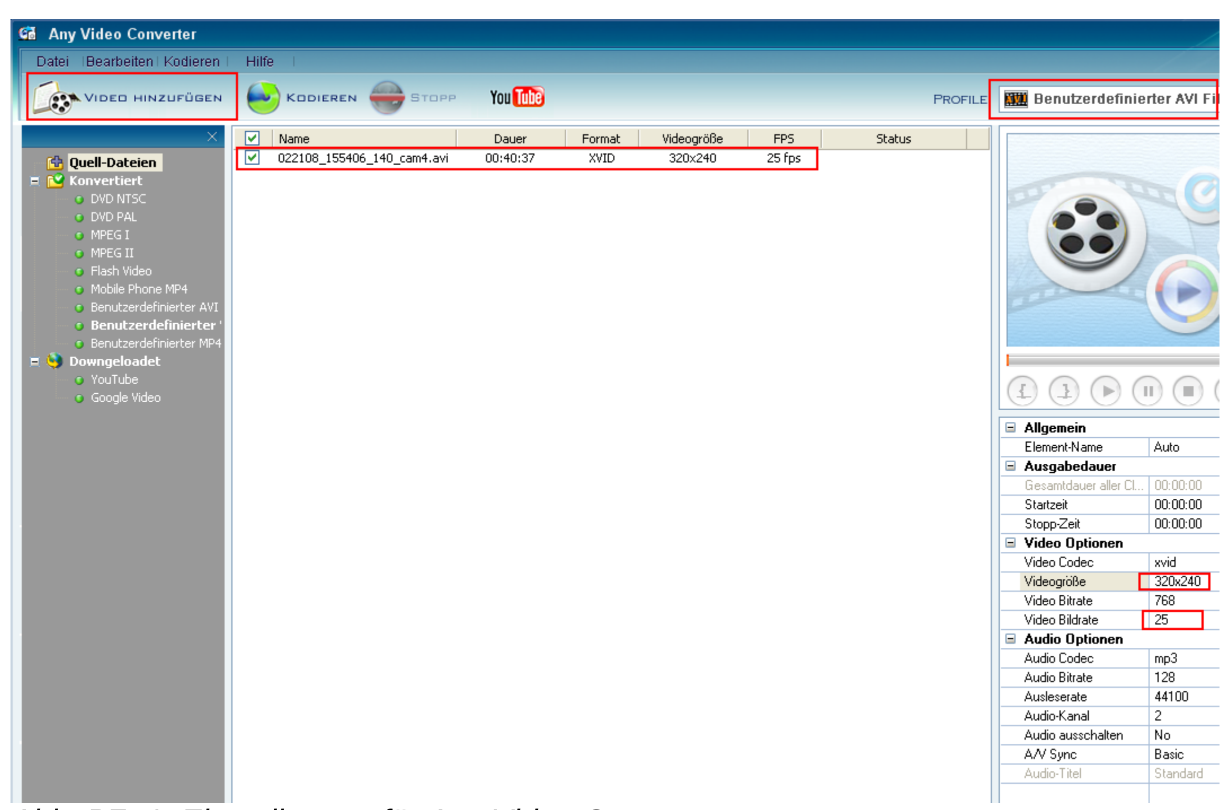


Abb. B7- 1: Einstellungen für Any Video Converter

2. Sind alle Dateien geladen, über das Auswahlfeld Profile das gewünschte Ausgabeformat (z.B. AVI oder WMV) festlegen, in das umgewandelt werden soll.
3. Über den Menüpunkt „Datei“ das Zielverzeichnis festlegen.
4. Über „Kodieren“ den Konvertierungsvorgang starten.

Anhang C – Abbildungsverzeichnis

Anhang D - Literaturverzeichnis

[Graf 2006] C. Graf: Emotionen in der Mensch-Maschine-Interaktion,
Otto-von-Guericke-Universität Magdeburg, 2006

[Nelius 2003] M. Nelius: Mousemap-basierte Erkennung der
Problemfelder von Anwendern bei der Bedienung von Software,
Universität Rostock, 2003

[Nielsen 1994] Jacob Nielsen, Usability Engineering, Morgan Kaufman,
New Edition, 1994

[Richter 2008] Stefan Richter, Bereitstellung eines Eyetracking-Systems
für Lehre und Forschung, Diplomarbeit HAW Hamburg, 2008

[Schwarze 2003] André Schwarze: Konzept und Einrichtung eines
Ergonomie-Labors, Diplomarbeit HAW Hamburg, 2003

[Ulas 2005] Bülent Ulas, Fatih Keles: Mouse Capturer Benutzerhandbuch,
Arbeit im Projektpraktikum Usability-Engineering
an der HAW Hamburg, 2005

[XviD 2006] C. Lampert, P. Massimino, M. Militzer, P. Ross: XviDCodec,
http://www.xvid.org, 2006

[Wandmacher 1993] J. Wandmacher: Software-Ergonomie,
Gruyter Verlag Berlin, 1993

[Wolf 2006] Daniel Wolf: Bereitstellung effizienter Auswertungsmethoden
von Usability-Tests sowie prototypische Entwicklung eines
Analysewerkzeuges, Bachelorarbeit HAW Hamburg, 2006

[Yaylacioglu 2007] Metin Yaylacioglu: Usability-Labor-Netzwerk
(Netzwerkhandbuch), Arbeit im Projektpraktikum Usability-Engineering
an der HAW Hamburg, 2007

[Zlotopolski 2006] Mark Zlotopolski, Tobias Wittmaack:
BuzzerTool Dokumentation, Arbeit im Projektpraktikum Usability-Engineering
an der HAW Hamburg, 2006

Zusätzlich: diverse Bedienungsanleitungen der neu beschaffenen Hardware für das
Usability – Labor

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §24(5) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, den 19.06.2008

--	--------------------------------------
Ort, Datum	Unterschrift